Sales Part Number M-MPU28
Manufacturing Part Number M7002
June 8, 1990
Rev B

SBE MPU-28

68020 *Multibus-Compatible Single-Board Computer

User Reference Manual

IMPORTANT: If you are currently using the documented product, fill out the enclosed pink card and return it to SBE to receive updates to this manual. The information you fill in will enable SBE to send the updates directly to you.

SBE, Inc.
2400 Bisso Lane
Concord, California 94520-4804
(415) 680-7722
FAX 415-680-1427 (800) 347-2666

Reprints of all or part of the following manuals have been included by permission:

AMD, AmZ8030/AmZ8530 Serial Communications Controller Technical Manual, (March 1986) Copyright (c) 1986 Advanced Micro Devices, Inc. Reprinted with permission of copyright owner. All rights reserved. Reproduced by permission. 1986 Zilog, Inc. This material shall not be reproduced without the written consent of Zilog, Inc.

Motorola, MC68230 Parallel Interface/Timer (PI/T) "Advance Information", (December, 1983)

Motorola, MC68881/MC68882 Floating-Point Coprocessor User's Manual, (SBE adaptation of Section 1)

NCR, NCR 53C90 Enhanced SCSI Processor Data Sheet, (November, 1987)

LIST OF CHANGES TO THIS MANUAL

Rev A: Initial Release

How Rev B differs from Rev A:

1. The Rev B manual contains a new set of programming examples. These examples eliminate serial port initialization tables showing the use of autovectored interrupts since the MPU-28 does not support autovectored interrupts for serial port initialization.

2. The address of the watchdog timer is $03F80F01. The Rev A manual was incorrect and has been changed in Rev B.

3. Section 3-1 is restructured, categorizing static memory device sockets into banks 0 and 1.

4. The parts list for the MPU-28 has been updated.

5. The PARW bit in the control/status register is 0 for odd parity and 1 for even parity. The Rev A manual was incorrect and has been changed in Rev B.

6. Section 3-4-1 is reorganized separating receiving and generating Multibus interrupts.

7. Jumper J46, formerly used for configuring memory, is hardwired and is no longer a customer option. Memory options are still discussed, however, for customers who have earlier versions of the MPU-28 with this jumper option.

CONVENTIONS USED IN THIS MANUAL

Low active signals are indicated with an asterisk (*) after the signal name.

Registers illustrated in this manual associate a mnemonic with each bit; bit 0 is the least significant bit of each byte. Register bits called "unused" are bits that do not affect the function of the register, and, if read, yield no significant information.

A lower-case "x" denotes one of many variables. For example, INTx represents INT bits 0-7.

Throughout this manual, MPU-28 component locations are identified assuming the MPU-28 is facing the direction as shown below.

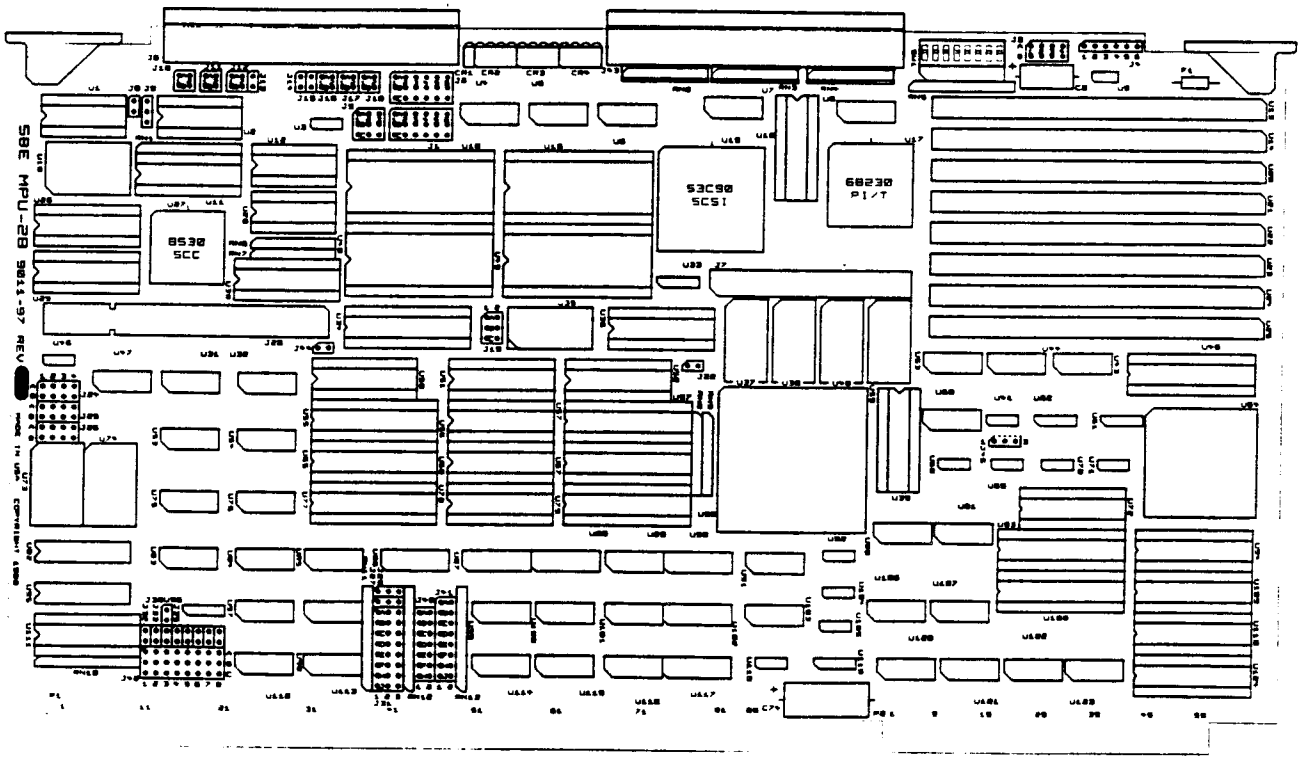


Figure I: MPU-28 Board Orientation

Table Of Contents

List Of Figures

LIST OF FIGURES

------ NOTES ------

SBE MPU-28

68020 Multibus-Compatible
Single-Board Computer

User Reference Manual

1. PRODUCT DESCRIPTION

The SBE MPU-28 board is a high-performance Multibus-compatible processor board that features the 32-bit 68020 microprocessor. With 1, 2, 4, or 8Mbytes of on-board shared memory, the MPU-28 runs at 12.5MHz with no wait states or 20MHz with 1 wait state. A 68881 or 68882 floating-point math coprocessor can be added to further enhance the MPU-28's computing power.

The board's features include:

* Parity protection of on-board DRAM
* Four 28-pin EPROM sockets (two of which may contain static RAM, battery-backup RAM, EPROM, or EEPROM)
* On-board SCSI support
* On-board memory management
* One 8-/16-bit iSBX* connector
* Two multiprotocol full-duplex serial ports
* A 24-bit parallel port and timer
* Eight user-programmable LEDs
* Eight Software-Readable Switches
* "Mailbox" that allows other Multibus masters to reset, interrupt, and monitor the MPU-28.

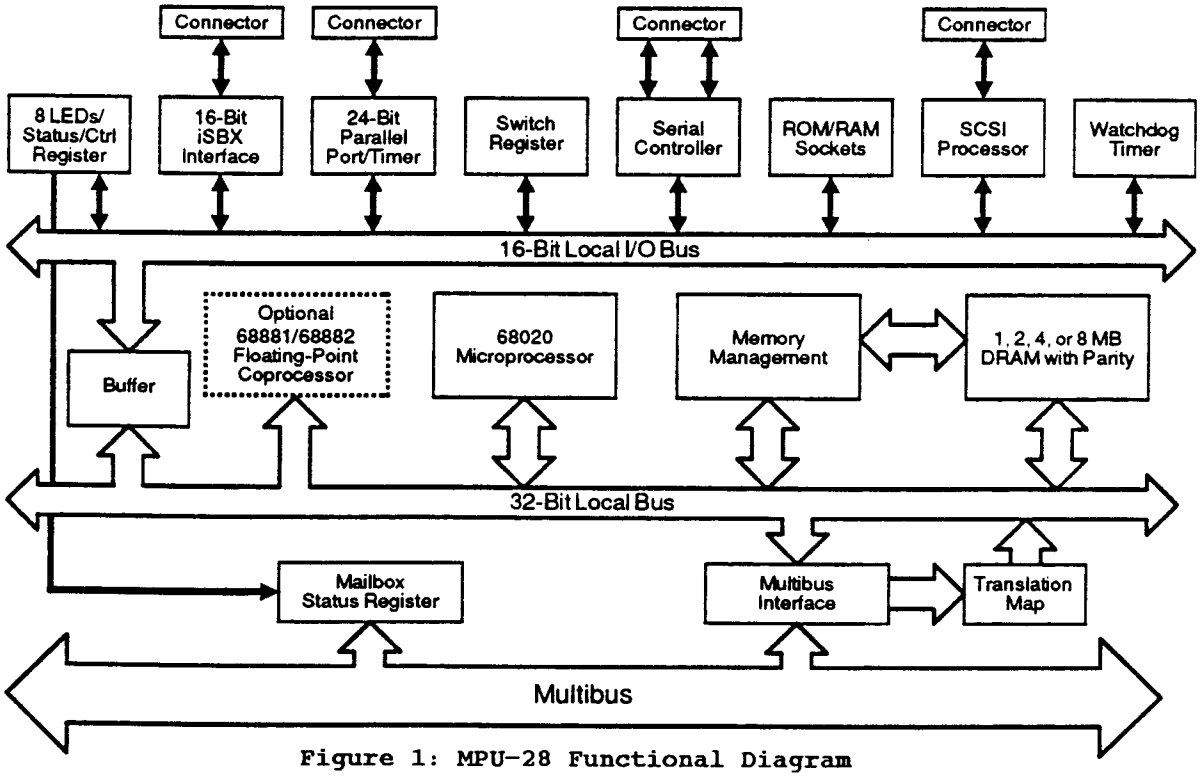


Figure 1: MPU-28 Functional Diagram

The MPU-28 also features the SBE Watchdog Timer which can help protect a system from major consequences of a malfunction. This fault timer can be programmed to halt a system by allowing programs on the MPU-28 to detect when normal processing has discontinued.

The MPU-28 is designed for use as a stand-alone microcomputer, a single CPU/controller in a Multibus system, or one of many CPU elements in a multiprocessor configuration. Able to accommodate numerous I/O and memory products compatible with Multibus and iSBX standards, and giving superior performance and ease of programming of the 68020 microprocessor, the MPU-28 is well suited to a wide range of challenging applications -- from data acquisition, to real-time image processing, to robotics.

Software available for the MPU-28 includes:

- SBE-UX operating system (an SBE derivative of UNIX* System V)
- REGULUS* operating system
- VRTX32/RTscope* versatile 32-bit real-time executive
- PROBUG® software debugger/monitor

1-1. Handling Procedures

Mishandling the MPU-28 can damage parts on the board or the MPU-28 itself. To avoid damaging this product, please observe the following precautions:

* Avoid any possibility of static electric discharge around the board.

* Keep the board in a conductive plastic bag or in contact with conductive foam when not installed in a system.

* Any equipment used to work on this board should be grounded. Any person handling the bare board should be grounded as well.

* Do not attempt to straighten any part soldered to the board as pin breakage or internal part damage might occur.

* Do not insert or remove the board while power is applied to the card cage.

* Do not insert or remove any devices while power is applied to the board.

* Do not apply external voltages to any devices on this board with power removed from the board.

1-1-1. Unpacking Instructions

If the carton containing the MPU-28 board appears to be damaged when you receive it, request that the carrier's agent be present when you unpack and inspect the equipment. After unpacking, verify that all items in the packing list are present. Inspect for any shipping damage to the equipment. Save all packing material for storing or reshipping the equipment. For repairs or replacement of equipment damaged during shipment, contact SBE, Inc. to obtain a Return Materials Authorization number and further shipping instructions.

2. SPECIFICATIONS

2-1. Speed Of Operation

Processor	12.5 or 20MHz 68020
CPU Processor Clock Accuracy	±0.01%
Bus Clocks	BCLK and CCLK - 100ns cycle times
Memory Cycle Time (on-board)	DRAM: 160ns @ 12.5MHz 200ns @ 20MHz
	EPROM: Wait states depend on speed of EPROMs

2-2. Memory Capacity

EPROM	Sockets for up to 256Kbytes
DRAM	1, 2, 4, or 8Mbytes
Static RAM	Sockets for up to 64Kbytes
EEPROM	Sockets for up to 64Kbytes

NOTE: Socket devices come in the following sizes:

EPROM Size	SRAM	EEPROM
8Kbytes	16Kbytes	16Kbytes
16Kbytes	64Kbytes	16Kbytes
32Kbytes	256Kbytes	64Kbytes
64Kbytes		

2-3. CPU

SBE Mailbox Allows other Multibus masters to interrupt or reset the MPU-28 or read status bits.

SBE Watchdog Jumper-selectable timeout: 1/10 to 13 seconds.

2-4. Input/Output

Serial I/O Two multiprotocol lines using one 8530 Serial Communications Controller (SCC) or equivalent. Each channel, equipped with RS-232-C drivers and receivers, can be independently jumpered for DTE or DCE operation.

Parallel Interface/Timer
One 68230 Parallel Interface/Timer or equivalent; 24 bits parallel I/O.
Timer can be used to generate periodic system interrupts.

SCSI Controller	One 53C90 Enhanced SCSI Controller or equivalent provides on-board SCSI support.
Floating-Point Coprocessor	One optional 68881/68882 Floating-Point Coprocessor; 80-bit floating-point coprocessor calculations.

2-5. Connectors

J4	Reset/NMI connector
J6	50-pin RS-232-C dual serial port header connector
J7	34-pin Parallel port header connector
J20	44-pin iSBX connector accepts an 8- or 16-bit, single- or double-width iSBX module.
J43	SCSI 50-pin connector
P1	Multibus connector
P2	Multibus connector

2-6. LEDs & Switches

Red LEDs	8 red software-controlled status LEDs 1 red MPU-28 power indicator 1 red status indicator for fuse-protected +5V 1 red PARITY ERROR indicator 1 red FAULT indicator
Green LED	1 green HALT indicator
Switches	Eight software readable switches

2-7. Serial Communication

Serial Line Drivers

The MPU-28 has RS-232-C line driver/receiver IC components on-board.

Asynchronous Operation

5- to 8- bit character
odd, even, or no parity
1, 1.5 or 2 stop bits
Error detection: framing, overrun, and parity

Byte Synchronous Operation

One or two sync characters
Automatic CRC generation and checking (CCITT-16)
IBM Bisync compatible in ASCII and EBCDIC coding formats

Bit Synchronous Operation

SDLC/HDLC flag generation and recognition
8-bit address recognition
Automatic zero bit insertion and deletion
Automatic CRC generation and checking (CCITT-16)

Encoding FM, NRZ, NRZI encoding

Baud Rates Independently selectable for each channel
External: 0 baud to 800Kbits/second

Programmable:
Asynchronous. . . . 50 baud to 38.4K baud
Synchronous. . . . 28 baud to 64Kbits per second

2-8. Multibus/IEEE 796 Compliance

Master. . . . . D16 M24 I16 VOL
Slave D16 M24

2-9. Physical Characteristics

Width . . . . . 12.00 in. (30.5 cm)
Height. . . . . 6.75 in. (17.2 cm)
Weight. . . . . 21.50 oz. (609.5 g)

2-10. Operating Requirements

Climate
Storage Temperature. . . . . -40 to +85 degrees C
Operating Temperature. . . . . 0 to 70 degrees C (at board surface)
Relative Humidity. . . . . 0 to 95% (noncondensing)

Environment: This equipment is designed to be used and stored in a commercial environment. Exposure to contaminating and/or corroding environments, such as salt water or excessive dust, is damaging to this equipment and is not covered under the warranty.

2-11. Power Requirements

Power. . . . . +5V ±5% at 8A typical
+12V ±5% at 25 mA minimum
-12V ±5% at 25 mA minimum

(NOTE: Assumes no iSBX modules)

CAUTION: Boards should be brought to operating temperature in a noncondensing environment. The rate of change in board temperature should not exceed 2 degrees C per minute.

2-12. Ordering Information

Figure 2-12a includes MPU-28 options. Some combinations of options are readily available while others may require additional production time. Contact the SBE Sales Department for specific ordering information, option availability, or custom services.

Figure 2-12a: MPU-28 Options

Feature	Options	Comments
Processor Speed	12.5MHz 20MHz	
Memory	1MB 2MB 4MB 8MB	An MPU-28 with 2 or 8Mbytes of memory fills the space of two Multibus slots except when placed in the first slot of most popular card cages.
Floating-Point Coprocessor?	yes no	If yes, select either a 68881 or 68882 Floating-Point Coprocessor.
Memory Management?	yes no	An MPU-28 with memory management can run in MMU or non-MMU mode which is selected under software control.

Figure 2-12b lists all supporting products for the MPU-28. User reference manuals automatically accompany those products that are ordered.

Figure 2-12b: Supporting Products For The MPU-28

Product Type	Product Name	Description
iSBX Modules	SASX	iSBX-SASI interface module (with variable length reset pulse)
	SCSI	iSBX-SCSI interface module (5380 controller)
	SCSI-CP	iSBX-SCSI interface module (with clock and printer interface)
	SERX-232	Dual-serial RS-232-C module
	SERX-422	Dual-serial RS-422 module
Cables	Serial Cable	Dual serial cable
Software	REGULUS	REGULUS operating system
	FORTRAN	FORTRAN 77 for REGULUS
	SYSTEM III	SYSTEM III Utilities for REGULUS
	*SBE-UX	SBE-UX operating system
	PROBUG	PROBUG software debugger/monitor
	VRTX32/RTscope	VRTX32/RTscope versatile 32bit real-time executive

* SBE-UX is an SBE derivative of UNIX System V, Release 2.0.

------ NOTES ------

3. OPTIONS AND JUMPER SETTINGS

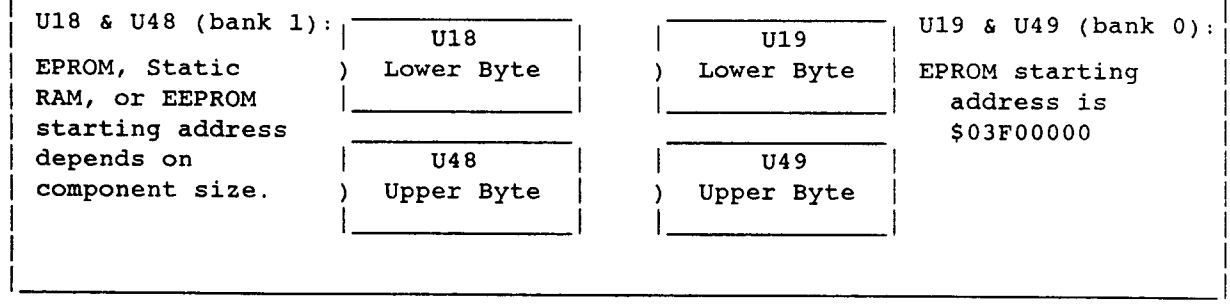
When you receive the MPU-28, it is jumpered for various defaults. For example, the standard jumper configurations for 28-pin EPROMs set a default speed of 300-400 nanoseconds and a default size of 16K x 8 bits. In its standard configuration, the board is a primary master on the Multibus using serial priority (see the standard jumper configuration drawing). Sections 3 and 4 describe how to change the MPU-28's default jumper settings.

3-1. 28-Pin Sockets For Static Memory Devices

The MPU-28 has four 28-pin sockets, U18 & U48 and U19 & U49, which hold either 24- or 28-pin memory components. For the board to operate when power is applied, EPROMs that include initialization data for the 68020 must go into sockets U19 and U49. (See Section 7-7 for details.) SBE's software debugger/monitor PROBUG is recommended if your board is for program development applications. The other two sockets, U18 and U48, may contain additional EPROMs, static RAMs, or EEPROMs.

The memory components that fit into the 28-pin sockets are 8 bits wide and can be accessed in pairs to form a 16-bit word. Upper bytes (bits 8 through 15) of 16-bit words are assigned to sockets U49 and U48. Lower bytes (bits 0 through 7) of 16-bit words are assigned to sockets U18 and U19.

Figure 3-1: Socketed Memory Starting Address



For the bank 0, socket device size is controlled by J5. For bank 1, size and type (EPROMs, static RAMs, or EEPROMs) are controlled by J1 and J2. The base address of the devices in bank 1 is governed by J3 which must be set for the largest component in banks 0 and 1. The speed of the four sockets is controlled by jumper area J24. These jumper areas are discussed in the following sections.

In the standard configuration, bank 0 (U19 and U49) makes up 16-bit words in the address range $03F00000-$03F03FFF. Likewise, bank 1 (U18 and U48) makes up the lower and upper bytes of 16-bit words in the address range of $03F04000-$03F07FFF. See the figure in the following section.

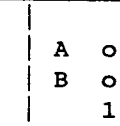
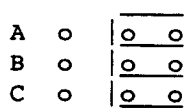
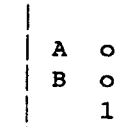
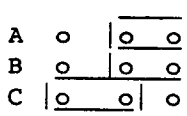
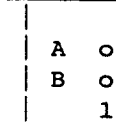
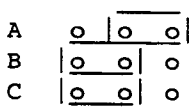
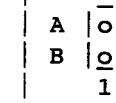
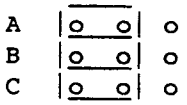
3-1-1. Setting 28-Pin Memory Component Size

J3 must be jumpered to match the size of the largest-capacity component installed in U18, U48, U19 and U49. The size you select governs both pairs of memory sockets. In the standard configuration, J3 is set for 16K x 8-bit components.

Jumper area J5 connects pins 1, 26, and 27 of the bank 0 sockets to +5V or to address lines depending on the size of the components. See Figure 3-1-1a.

Note: Figure 3-1-1a assumes that all devices are the same size. If bank 1 devices are a different size than those in bank 0, jumper J3 for the larger device and jumper J5 for the size of the device in bank 0. For example, with 16K x 8 EPROMs in bank 0, the bank 1 address range for 8K x 8 SRAMs is $03F08000-$03F0BFFF.

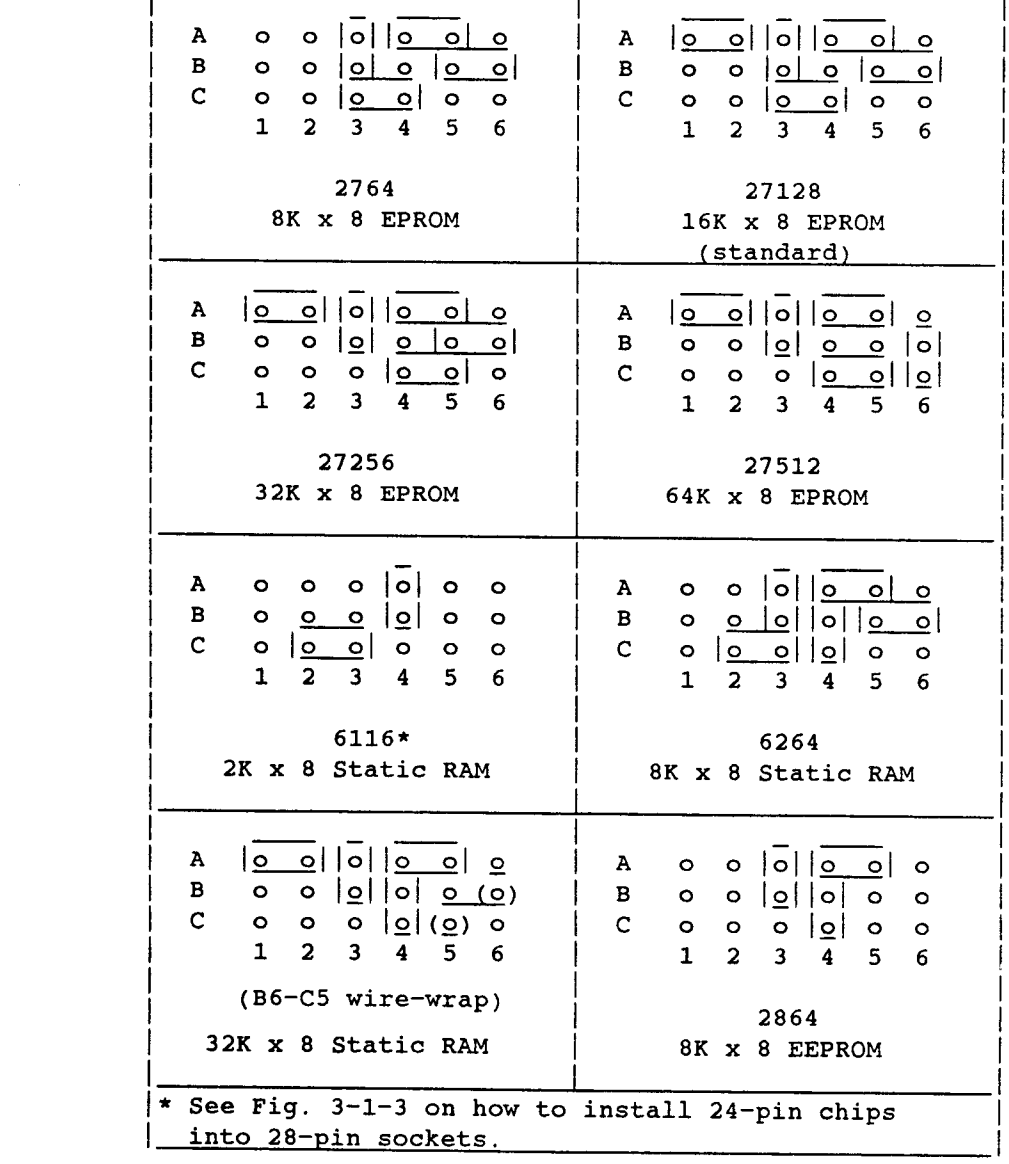
Figure 3-1-1a: Jumpering J3 and J5 For Memory Component Size

Largest Component Size	J3	J5	U19 and U49 Address Range (bank 0)	U18 and U48 Address Range (bank 1)
8K x 8			$03F00000 to $03F03FFF	$03F04000 to $03F07FFF
16K x 8 (standard)			$03F00000 to $03F07FFF	$03F08000 to $03F0FFFF
32K x 8			$03F00000 to $03F0FFFF	$03F10000 to $03F1FFFF
64K x 8			$03F00000 to $03F1FFFF	$03F20000 to $03F3FFFF

3-1-2. Jumpering for Various EPROM/RAM Types

Jumper areas J1 and J2 are located at the top edge of the MPU-28 between the header connectors. J1 and J2 are for configuring sockets U48 and U18 for the type of memory devices to be installed. The MPU-28 supports EPROMs (8K x 8 to 64K x 8), static RAMs (2K x 8, 8K x 8, and 32K x 8), and EEPROMs (8K x 8, 16K x 8, and 32K x 8). J1 and J2 are identical jumper areas and should both be set according to the following chart. If no memory components are used in U18 and U48, no jumpers need to be installed in J1 and J2.

Figure 3-1-2
Jumpering J1 And J2 For Various EPROM/RAM Types



* See Fig. 3-1-3 on how to install 24-pin chips into 28-pin sockets.

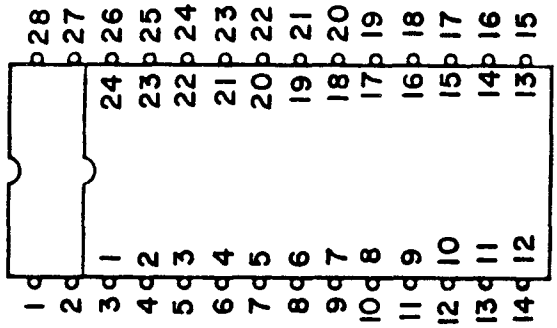
Jumper areas J1 and J2 are set up to support power-fail battery backup for volatile memory components installed in sockets U18 & U48. Pin B2 of each jumper area is connected to finger 2 (PBKUP) of the Multibus P2 connector. This line is intended to be used for battery backup. Pin C1 of J1 and J2 is connected to finger 1 (PDN) of the P2 connector. This line is intended to signal a falling voltage condition to external power-fail protection circuitry.

3-1-3. How To Install 24-Pin Components In 28-Pin Sockets

To install 24-pin memory components in the 28-pin sockets, orient the MPU-28 so that the Multibus P1 and P2 connectors are at the bottom of the board. Install the 24-pin component so that the end without the notch is lined up with the right edge of the socket, placing pin 1 of the component in pin 3 of the socket.

CAUTION: DO NOT MATCH PIN 1 OF THE COMPONENT WITH PIN 1 OF THE SOCKET IT MAY CAUSE DAMAGE TO THE MEMORY COMPONENT.

Figure 3-1-3: Position Of 24-Pin Component In 28-Pin Socket



3-1-4. Setting EPROM Access Time

Jumper area J24 selects EPROM access time for the 28-pin sockets. In the standard configuration, J24 is set for 350nsec EPROMs at 12.5MHz and 200nsec EPROMs at 20MHz.

Since the jumpering for access time affects all four EPROM sockets, we suggest that you select EPROMs for sockets U18 and U48 that match the speed of the components in U19 and U49 fairly closely. If the speeds of the two pairs of memory components do not match, the board must be jumpered for the slower of the two types.

Figure 3-1-4a shows how to jumper J24 for EPROM access times running at 12.5MHz.

Figure 3-1-4a: J24 Jumpering For EPROM Access Time At 12.5MHz

Jumper Area J24	# Of Wait States	Required EPROM Speed
A o o o o B o o o o 1 2 3 4	1 wait state	150nsec
A $\overline{o}$ o o o B o o o o 1 2 3 4	3 wait states	300nsec
A o $\overline{o}$ o o B o o o o 1 2 3 4	4 wait states (standard)	350nsec
A o o $\overline{o}$ o B o o o o 1 2 3 4	5 wait states	450nsec
A o o o $\overline{o}$ B o o o o 1 2 3 4	6 wait states	500nsec

Figure 3-1-4b shows how to jumper J24 for EPROM access times running at 20MHz.

Figure 3-1-4b: J24 Jumpering For EPROM Access Time At 20MHz

Jumper Area J24	# Of Wait States	Required EPROM Speed
A o o o o B o o o o 1 2 3 4	1 wait state	70nsec
A $\overline{o}$ o o o B o o o o 1 2 3 4	3 wait states	150nsec
A o $\overline{o}$ o o B o o o o 1 2 3 4	4 wait states (standard)	200nsec
A o o $\overline{o}$ o B o o o o 1 2 3 4	5 wait states	250nsec
A o o o $\overline{o}$ B o o o o 1 2 3 4	6 wait states	300nsec

3-2. Dynamic RAM Options

The MPU-28 is available in four DRAM versions. Two versions support 256Kbit DRAM chips for a total on-board memory of 1 or 2Mbytes. The other two versions support 1Mbit DRAM chips for a total on-board memory of 4 or 8Mbytes.

As shown in the following figure, four MPU-28 memory capacities are possible. Jumper J46 and the PAL in U39 determine whether 256Kbit or 1Mbit DRAM chips are used. The standard memory configuration is for 1Mbit DRAM chips.

Figure 3-2: MPU-28 Memory Capacities

J46	DRAM Size	Total DRAM	Highest Memory Address	U39 PAL Part #
o |o o| 1 2 3	256K x 9	1MB	$0FFFFF	66294
	256K x 9	2MB	$1FFFFF	
|o o| o 1 2 3 (standard)	1Mbit x 9	4MB	$3FFFFF	66293
	1Mbit x 9	8MB	$7FFFFF	

NOTE: MPU-28 boards with 2 or 8Mbytes of memory fill the space of two Multibus slots except when placed in the first slot of most popular card cages. In that case, only one Multibus slot is required.

3-3. Control/Status Register

The MPU-28's control/status register consists of four bytes used to control Multibus interrupts, watchdog timer enable, programmable red status LEDs, translation map selection, and parity functions. In addition, the control/status register contains read-only bits associated with software-readable switches. All bits of the control/status register are readable without affecting the status conditions.

Reading Byte 0 of the control/status register returns bits indicating the status of the iSBX interrupts plus four other MPU-28 status conditions. Writing to Byte 0 controls the eight programmable red LEDs. Bit assignments for Byte 0 are as follows:

Figure 3-3a: Bits Of The Control/Status Register, Byte 0
ADDRESS: $03F80900

Action	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read	MISTAT	PERR	LINSEL	ENBTF	DREQ	SCSIRQ	MINTR1	MINTR0
Write	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED 0

MISTAT — This bit indicates the status of Multibus interrupts. If MISTAT is 0, the MPU-28 is not generating an interrupt to the Multibus. If MISTAT is 1, and J31 is appropriately jumpered, the MPU-28 is currently asserting a Multibus interrupt.

PERR — When a parity error is detected in memory, this bit is set to 1 and the parity error interrupt is generated. PERR is returned to 0 and the interrupt is cleared using RPERR*. PERR is set to 0 on power-up or reset and is read only.

LINSEL — LINSEL indicates whether a mailbox interrupt is pending. When another Multibus master writes data to the MPU-28's mailbox and the mailbox interrupts are enabled (RINSEL* bit is not active), LINSEL is set to 1 and an interrupt is generated to the MPU-28. See sections 3-4 and 3-5.

This read-only bit is cleared to 0 at power-up, reset, or when RINSEL* is set to 0.

ENBTF — This bit is cleared to 0 at power-up or reset. It is set to 1 when the watchdog timer is activated via a write of a byte containing $2A at location $03F80F00. This bit and ENBTF in the Mailbox Status Register always reflect the same condition. ENBTF is read only.

DREQ — This bit indicates a pending DMA request from the SCSI. If this bit is set to 1, bytes can be read from or written to the SCSI's FIFO. If a 0, bytes cannot be read from or written to the SCSI's FIFO. Upon reset or power-up, DREQ is cleared to 0. Use of DREQ depends on the use of the SCSI's DMA-mode commands. For more information, see the NCR 53C90 manual at the end of this manual.

SCSIRQ — This bit allows the system's interrupt processing routine for Level 2 interrupts to detect if the on-board SCSI interrupt is active. A 1 value for this bit represents an active interrupt. The bit is cleared by servicing the interrupt condition or when the MPU-28 is powered up or is reset. SCSIRQ is read only.

MINTR0-MINTR1 — These bits allow the system's interrupt processing routine for Level 2 interrupts to detect which, if any, of the two iSBX interrupts are active. A 1 value for any of these bits represents an active interrupt. The bit is cleared by servicing the interrupt condition. MINTR0-MINTR1 are read only.

NOTE: MINTR0 and MINTR1 have values of 1 when there is no iSBX module installed on the MPU-28; however, no interrupt is generated unless an iSBX module is present. These bits are set to 1 at power-up or reset.

LED7-LED0 — The programmable red LEDs on the edge of the board are controlled by these bits. Setting a bit to 0 causes its corresponding LED to be lit; setting the bit to 1 will cause the LED to be turned off. LED7 is the red LED closest to the green LED. These are write-only bits. These bits are set to 0 upon power-up or reset.

Byte 1 of the control/status register holds eight read/write status bits. Individual bits of Byte 1 can be altered by performing a read to obtain the current byte pattern, changing the desired bit, and writing the resulting pattern back to Byte 1.

Bit assignments for Byte 1 are as follows:

Figure 3-3b: Bits Of The Control/Status Register, Byte 1
ADDRESS: $03F80901

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
READY	MS1	MS0	BS	RPERR*	PARW	GENIRQ	RINSEL*

READY — This bit primarily serves as a status indicator to other Multibus masters. Cleared to 0 on reset or power up, the READY bit can be set by the MPU-28 system (or alternately set and cleared) to indicate any user-defined condition. For example, a program on the MPU-28 could set the bit to 1 after completing start-up diagnostics and initialization.

Other Multibus masters can be programmed to poll the status of the READY bit (see the description of the "Mailbox Status Register") to synchronize accesses to the MPU-28 with appropriate READY conditions.

MS1 & MS0 These bits provide the two most significant bits of the 10-bit input to the address translation scheme for off-board accesses to the MPU-28's shared memory. Since these bits are controlled entirely by the MPU-28 program, they allow instantaneous, transparent address map switching among up to four different translation maps.

This is useful, for example, when it is necessary to alter several locations in a map at one instant: the original map can be left unaltered while setting up an alternate. The new map is then selected by the MPU-28 with one write instruction which switches maps via MS0 and MS1. See Section 4-7 for more information regarding setup and control of address translation maps.

Both MS0 and MS1 are cleared to 0 on power up or reset.

BS This bit provides one method of program control of byte swapping for MPU-28 accesses to Multibus memory. If a jumper is installed connecting pins 2 and 3 of J28, the status of BS controls byte swapping. Otherwise, the status of BS has no effect.

When active, clearing BS to 0 causes byte swapping on all Multibus memory accesses. A 1 in BS results in unswapped accesses. BS is cleared to 0 on power-up or reset.

RPERR* Setting this bit to 0 clears the parity status bit (PERR, control/status register Byte 0) to 0 (inactive) and clears the parity interrupt. This bit must be set back to 1 to enable detection of parity errors in memory. This bit is cleared to 0 on power-up or reset. Leaving RPERR* set to 0 disables parity interrupts.

PARW This bit determines the parity (odd or even) that is generated by writes to memory. PARW is 0 for odd parity and 1 for even parity. PARW is used only for parity logic diagnostics. It is set to 0 on power-up or reset and should not be changed for normal operation.

GENIRQ This read/write bit can be used to generate interrupts to the Multibus. Its effect depends on the jumper configuration of J9. See Section 3-4-1.

When J9 is jumpered pin 1 to pin 2:
Writing to GENIRQ asserts or negates an interrupt to the Multibus. Writing a 1 generates an interrupt on the Multibus while writing a 0 negates the interrupt. Reading this bit determines whether the MPU-28 is generating an interrupt to the Multibus.

When J9 is jumpered pin 2 to pin 3:
Writing to GENIRQ generates a Multibus interrupt or causes the interrupt to remain unchanged. A transition from 0 to 1 generates a Multibus interrupt. A transition from 1 to 0 has no effect. The only way to clear the interrupt is through the MPU-28's mailbox. To generate a new interrupt, a 0 must be written and then a 1. GENIRQ is set to 0 on power-up or reset. Reading this bit simply shows the last value written to it which is different from the

MISTAT bit (described earlier in this section) which always reflects the current interrupt status. You should, therefore, read MISTAT rather than GENIRQ to determine the status of Multibus interrupts.

RINSEL* Setting this bit to 0 clears the mailbox status bit (LINSEL) to 0 (inactive) and clears the mailbox interrupt. This bit must be set back to 1 to enable detection of mailbox interrupts. Leaving RINSEL* set to 0 disables mailbox interrupts.

NOTE: Control/status register bytes 0 and 1 can be read together using a word instruction to facilitate checking the current status of all 16 bits. While word length writes can also be performed, keep in mind that writes to Byte 0 control the programmable LEDs. Therefore, writes to bytes 0 and 1 are usually performed using byte-length instructions.

Byte 2 of the control/status register is used to read the current status of eight inputs from jumper area J31. J31 can be jumpered so that any or all of the Multibus interrupt lines INT0*-INT7* are readable via Byte 2. (See Section 3-4-1.)

Reading a 0 for a bit in Byte 2 indicates that no active interrupt is being received from the corresponding Multibus interrupt line. Reading a 1 indicates an active Multibus interrupt. Bits of control/status register Byte 2 are assigned to INT0*-INT7* as shown in the following figure.

Figure 3-3c: Bits Of The Control/Status Register, Byte 2
ADDRESS: $03F80902

Action	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read	INT0*	INT1*	INT2*	INT3*	INT4*	INT5*	INT6*	INT7*
Write	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED 0

NOTE: As shown in Figure 3-3c, Byte 2 of the control/status register can also be written to. A write is equivalent to writing to Byte 0 and sets the programmable LEDs.

Byte 3 of the control/status register contains bits that read the status of the eight switches located at the edge of the MPU-28 board.

Bit assignments for Byte 3 are as follows:

Figure 3-3d: Bits Of The Control/Status Register, Byte 3
ADDRESS: $03F80903

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SWITCH 8	SWITCH 7	SWITCH 6	SWITCH 5	SWITCH 4	SWITCH 3	SWITCH 2	SWITCH 1

SWITCH 1-SWITCH 8 — These read-only bits indicate the settings of the eight switches accessible at the edge of the MPU-28 board. The eight switches are numbered 1-8 on the switch box. Setting a switch to the OPEN position causes the corresponding bit in the control/status register to be read as a 1. Likewise, setting a switch to the CLOSE position causes the corresponding bit in the control/status register to be read as a 0. These eight switches make it convenient to assign IDs or control program functions on multiple MPU-28 boards without using different EPROMs.

3-3-1. Parity Status And Control

The MPU-28's parity checking feature detects single bit errors during reads of the MPU-28's dynamic RAM. To do this, it calculates and saves the parity of each byte written to memory. When data is written in 16-bit words or 32-bit long words, a parity bit is saved for each byte stored. The MPU-28 should be configured to calculate odd parity because at power-up or reset the PARW bit in byte 1 of the control/status register is set to odd parity. PARW, however, can be set to even parity for diagnostic purposes. The PARW bit controls both parity verification and generation.

When data is read back from memory, the parity checking circuitry calculates the parity bit for each byte and compares this value against the previously stored parity bit. If a mismatch is detected and RPERR* is 1, the PERR bit in the system control register is set to 1 and an interrupt is generated.

During single byte, word, or long word accesses, the parity checking circuitry uses the parity of each byte of the long word fetched by the 68020 microprocessor to determine if a parity error exists.

The parity status remains in PERR until explicitly cleared via the system control register. This is done by writing a 0 to the RPERR* bit. This resets PERR and clears the interrupt. To enable detection of another parity error, a 1 must be written to RPERR*.

NOTE: If enabled, parity checking takes place on all accesses of the MPU-28's dynamic RAM. Thus, a MPU-28 memory access by another master on the Multibus could cause a parity interrupt even though the MPU-28 is not accessing its own DRAM.

CAUTION: **Reading from a location that has never been written to can cause parity errors. It is recommended that you clear all of RAM memory as part of the board initialization procedure.**

3-3-2. LED Programming

Red LEDs. The four red status LEDs mounted farthest from the green LED consist of: an MPU-28 power indicator, a status indicator of the fuse-protected +5V, a parity error indicator, and a fault indicator.

Writing to Byte 0 of the control/status register controls the eight red LEDs located between the green LED and the four hardware-controlled red LEDs. Setting an LED bit to 0 will cause the corresponding LED to be illuminated; setting the bit to 1 will cause the LED to be off.

Green LED. One green status indicator is mounted along the top edge of the MPU-28 adjacent to the J6 header. It reflects the status of the HALT line. On within a second after power is applied to the board, the green LED stays on when the processor is operating and goes off and stays off when a HALT is asserted.

The only conditions that cause a HALT to be asserted on the MPU-28 are a watchdog timer fault (see Section 3-6) and a double bus fault (i.e., when a bus error or address error exception occurs during bus error or address error exception processing). The green LED also goes off momentarily after the board is reset by the reset button or through the mailbox.

The LED layout and functions are depicted in the following figures.

Figure 3-3-2a: LED Layout

Green LED	Eight Software-Controlled Red LEDs: Address = $03F80900							
GREEN	RED	RED	RED	RED	RED	RED	RED	RED
HALT	LED7	LED6	LED5	LED4	LED3	LED2	LED1	LED0
	Controlled by Control/Status Register Byte 0:							
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Figure 3-3-2b: LED Layout (con't)

Hardware-Controlled Red Status LEDs			
RED	RED*	RED	RED
POWER +5V to MPU-28	STATUS +5V to parallel port J7 connector pins 30, 32, & 34	PARITY ERROR	FAULT

* Associated with fuse F1

3-4. Interrupt Options

The sources of interrupts for the MPU-28 are the NMI button, parity logic, mailbox, iSBX connector, SCSI controller, two serial ports, parallel port, and the timer of the 68230 PI/T. Multibus interrupt lines INT0* to INT7* may also be used as interrupt sources.

The priority level of each of these interrupts is controlled by the PAL in socket U11. Level 7, the highest priority, is nonmaskable and is dedicated to connector J4. (See Section 7-6.) The parallel port, timer, and serial ports generate vectored interrupts. The other interrupt sources generate autovectored interrupts.

The standard priority assignments for the interrupts are:

Figure 3-4: Priority Of On-Board Interrupts

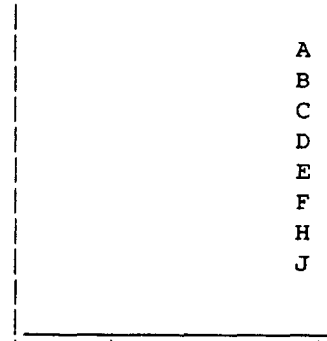
Type Of Interrupt	Priority Level
NMI Button	Level 7 (highest priority)
Parity Logic	Level 6
Two Serial Ports	Level 5
PI/T Timer	Level 4
Mailbox Interrupt	Level 3
Multibus INT0*-INT7*, iSBX Connector, and SCSI Controller	Level 2
PI/T Parallel Port	Level 1 (lowest priority)

As shown, an interrupt on Level 2 could come from one of the eight Multibus interrupt lines, one SCSI interrupt line, or two iSBX interrupt lines. The control/status register contains 12 bits indicating the status for each of these interrupt sources. Byte 0 (address $03F80900) contains four interrupt bits; two for iSBX and two for SCSI. Byte 2 (address $03F80902) contains the eight Multibus interrupt status bits. See Section 3-3 for more information.

3-4-1. Receiving And Generating Multibus Interrupts

The MPU-28 can receive as well as generate Multibus interrupts. The eight Multibus interrupt lines INT0* through INT7* are available on column 2 of J31, from A to J respectively (see Figure 3-4-1a). Jumpering any of these lines to the corresponding pin on column 3 of J31 allows the board to receive the interrupt. Receiving interrupts from the Multibus does not prevent use of the mailbox interrupt feature. (See Section 3-5 for more information.)

Figure 3-4-1a
J31: Receiving Multibus Interrupts



J31 Row	Generating	Multibus Interrupt Line	Interrupt Levels
	Column 1	Column 2	Column 3
A	-	INT0*	Level 2
B	-	INT1*	Level 2
C	-	INT2*	Level 2
D	-	INT3*	Level 2
E	-	INT4*	Level 2
F	-	INT5*	Level 2
H	-	INT6*	Level 2
J	-	INT7*	Level 2

The INTx bits in the control/status register are 0 for no interrupt and 1 for interrupt.
Note: No jumper installed in J32 means the MPU-28 will not receive the corresponding interrupt.

Interrupts received from the Multibus via J31 may or may not interrupt the MPU-28 depending on the setting of jumper J32. If a jumper is installed at J32, an interrupt received through J31 will interrupt the MPU-28.
Whether or not J32 is jumpered, a received Multibus interrupt causes a 1 to be read in Byte 2 of the control/status register. This allows the program on the MPU-28 to use polling for Multibus activity on INT0* through INT7*.

Custom PALs can be used at U111 to change the function of J32. For example, with an alternate PAL, J32 can be used to enable or disable selected groups of Multibus interrupt lines.

Figure 3-4-1b
J32: Received Multibus Interrupt Handling

Standard PAL U111	o o J32	Disabled: MPU-28 polls control/status register for interrupt
	o o J32	Enabled: MPU-28 interrupted

The MPU-28 generates Multibus interrupts when a jumper is installed on pins 1 and 2 of jumper area J31. Installing a jumper on these pins enables Multibus interrupt generation of a particular interrupt line. Writing to the GENIRQ bit (byte 1 of the control/status register) generates an interrupt if J31 is jumpered to drive an interrupt. Refer to Figure 3-3b. The status of interrupts on the Multibus can be read in the MISTAT bit in the control/status register. See Figure 3-3a for more information on MISTAT.

Figure 3-4-1c
J31: Generating Multibus Interrupts

J31 Row	Generating	Multibus Interrupt Line	Interrupt Levels
	Column 1	Column 2	Column 3
A	SYSIRQ	INT0*	-
B	SYSIRQ	INT1*	-
C	SYSIRQ	INT2*	-
D	SYSIRQ	INT3*	-
E	SYSIRQ	INT4*	-
F	SYSIRQ	INT5*	-
H	SYSIRQ	INT6*	-
J	SYSIRQ	INT7*	-

The INTx bits in the control/status register are 0 for no interrupt and 1 for interrupt.
Note: No jumper installed on J32 means the MPU-28 will not assert the corresponding interrupt.

The Multibus interrupt method is determined by setting jumper area J9. When J9 is jumpered 1 to 2, the interrupt is asserted and negated by the MPU-28. Changing the GENIRQ bit from 0 to 1 asserts the interrupt. Changing the GENIRQ bit from 1 to 0 removes the interrupt. See Section 3-3 for more information on these bits.

When J9 is jumpered 2 to 3, the interrupt is asserted by the MPU-28 and negated by the interrupted board. The interrupt is asserted when the GENIRQ bit makes the transition from 0 to 1. To generate another interrupt, the MISTAT bit **must** be set back to 0 first. Setting it back to 0 does not negate the interrupt. The interrupt is negated when another master on the Multibus writes a 1 to bit 1 of the mailbox. The MPU-28 can repeat the interrupt by writing to its own mailbox across the Multibus.

When J9 is jumpered 2 to 3, reading the MISTAT bit in the control/status register reveals the current status of the interrupt line. It will be a 1 when an interrupt is being asserted and a 0 when the interrupt has been cleared by the interrupted board via the MPU-28 mailbox.

Figure 3-4-1d: J9 Multibus Interrupt Method

Asserted & Negated by the MPU-28 via the GENIRQ Bit (standard)	o o o	1 2 3
Asserted by MPU-28 via GENIRQ Bit & Negated by the Interrupted Board via the Mailbox Address	o o o	1 2 3

3-5. SBE Mailbox

The SBE Mailbox circuitry allows other Multibus masters to interrupt and reset the MPU-28 and to clear Multibus interrupts generated by the MPU-28. Also, other Multibus masters can read the SBE Mailbox address and thereby monitor the status of up to eight MPU-28 conditions.

3-5-1. Mailbox Write Functions

The mailbox feature allows other Multibus masters to control functions on the MPU-28 by writing to its Multibus I/O address. Depending on the data written to the mailbox, a Multibus master may generate a local interrupt on the MPU-28, clear a Multibus interrupt that the MPU-28 is generating, or reset the MPU-28.

When a master on the Multibus writes a byte or word that has bit 0 set to 1 to the MPU-28's mailbox address, the LINSEL bit in the control/status register goes to 1. This in turn may be used to generate an on-board interrupt. (See RINSEL* and LINSEL in Section 3-3 for details.) This method may be used instead of, or in addition to, generating interrupts on a Multibus interrupt line.

To negate a Multibus interrupt asserted by the MPU-28, a master writes data with bit 1 set to 1 to the mailbox. J9 must be jumpered 2 to 3 for this option. (See Figure 3-4-1d.)

If a master on the Multibus writes zeroes to bits 0 and 1 of the MPU-28's mailbox address, the MPU-28 will perform a full hardware reset. This can be used by another master on the Multibus to return the MPU-28 to a known state. The mailbox reset will not hold the MPU-28 in a reset state, but will cause the board to restart as if the board has just been powered up.

The following figure summarizes the effect of bits 0 and 1 of data written by another master on the Multibus to the MPU-28's mailbox address.

Figure 3-5-1a: Mailbox Data Bit Functions

Upper Byte Of Data Written To MPU-28 Mailbox Address	Lower Byte Of Data Written To MPU-28 Mailbox Address*	Effect**
All bit settings are ignored.	xxxx xx00	Full hardware reset of MPU-28
	xxxx xx01	Generates mailbox interrupt
	xxxx xx10	Clears Multibus interrupt
	xxxx xx11	Clears Multibus interrupt and generates mailbox interrupt

* "x" indicates a bit whose setting is ignored.
** "Multibus interrupt" is from MPU-28; "mailbox interrupt" is to MPU-28.

NOTE: The data written to the MPU-28's mailbox address is not saved; it is only used to generate mailbox interrupts, to clear MPU-28-generated Multibus interrupts, or to reset the MPU-28.

The mailbox address resides in the 64Kbyte Multibus I/O address space. The address is jumper-selectable and may be either 8 bits or 16 bits. Jumper areas J27, J40, and J41 control these options.

Jumper area J27 allows you to select between an 8-bit and 16-bit mailbox address. With a jumper installed connecting pins 2 and 3, an 8-bit address is selected. With a jumper installed connecting pins 1 and 2, a 16-bit address is selected.

Figure 3-5-1b
J27: Selecting An 8-Bit Or 16-Bit Mailbox Address

8-Bit	16-Bit
	
1 2 3	1 2 3
	(standard)

If J27 is jumpered to select a 16-bit mailbox address, the bits corresponding to address lines A8 through A15 are set using J41. J41 has no effect if an 8-bit mailbox address has been selected.

Jumper area J41 is associated with address lines A15 through A8 of the Multibus I/O address space while jumper area J40 determines address lines A7 through A1 (not A0) of the mailbox address in the Multibus I/O address space. See the following figure.

Figure 3-5-1c: Multibus Address Lines Associated With J40 And J41 Pins

J40	Associated Address Line	J41*	Associated Address Line
		A o o	A15
A o o	A7	B o o	A14
B o o	A6	C o o	A13
C o o	A5	D o o	A12
D o o	A4	E o o	A11
E o o	A3	F o o	A10
F o o	A2	H o o	A9
H o o	A1	J o o	A8
1 2		1 2	

* Valid only if J27 is jumpered for 16-bit address

Installing a jumper causes the corresponding address bit to be a 1, whereas, no jumper installed causes a 0. For example, an 8-bit Multibus I/O address of $0A (0000 1010 in binary) for the mailbox interrupt would be jumpered as follows:

Figure 3-5-1d: Jumpering An 8-Bit Multibus Address Of $0A For The Mailbox Interrupt

J27	J40	J41
o |o o|	A o o B o o C o o D o o E |o o| F o o H |o o|	Doesn't Matter For 8-Bit Address
1 2 3	1 2	

3-5-2. Mailbox Status Register

The MPU-28's mailbox can be read by other Multibus masters allowing them to monitor the status of up to eight on-board conditions. The mailbox address is set up as described in Section 3-5-1.

The bits of the mailbox status register are assigned as shown in the following figure:

Figure 3-5-2: Mailbox Status Register Bit Assignments

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
READY	RESET	MISTAT	HALT	FAULT	PERR	LINSEL	ENBTF

Mailbox status register bits are read only. Writes to the mailbox are described in the previous section.

READY — This bit is cleared or set by the program running on the MPU-28. Writing a 0 to the READY bit (bit 7 in the control/status register at $03F80901) clears this bit in the mailbox status register to 0. Similarly, writing a 1 to the READY bit at $03F80901 sets this bit to 1.

The READY bit is useful for communicating the status of some user-defined condition to other boards on the Multibus. For example, the READY bit can be activated by the MPU-28 after completing diagnostics and initialization as part of a power-up or reset sequence.

READY will always be 0 following reset or power up.

RESET — This bit will be read as a 1 if and only if the RESET* pin of the 68020 is active. This includes resets generated by the 68020 reset instruction.

MISTAT — This bit is identical to the MISTAT bit (bit 7 in the control/status register at $03F80900). When MISTAT is 1, the MPU-28 is generating an interrupt. Jumper area J31 controls which of the Multibus interrupt lines is activated. By reading the status of MISTAT, another Multibus board can detect if the MPU-28 is the cause of an interrupt when it and some other device in the system are set up to interrupt on the same INT0*-INT7* line.

HALT — This bit reflects the current status of the 68020's HALT* pin. Reading a 1 indicates the 68020 is currently halted. There are only two conditions that will hold the HALT status bit at 1: a catastrophic failure resulting in a double bus fault, or the timeout of an activated watchdog timer. Double bus faults are detected and signaled by the 68020. A HALT caused by the watchdog timer can be differentiated from a double bus fault using the FAULT bit described next.

FAULT — This bit is cleared to 0 by a power-up or reset and is set to 1 only when an enabled watchdog timer times out. (See Section 3-6.)

PERR — This bit is equivalent to the PERR bit (bit 6 in control/status register at $03F80900). When a parity error is detected in memory this bit is set to 1. See Section 3-3 for a further description of PERR.

LINSEL — This bit is equivalent to the LINSEL bit (bit 5 in the control/status register at $03F80900). LINSEL is 1 if the Mailbox interrupt has been activated. See Section 3-3 for further information about LINSEL.

ENBTF — This bit is equivalent to the ENBTF bit, (bit 4 in the control/status register at $03F80900). When ENBTF is 1, the MPU-28's watchdog timer has been activated. (See sections 3-3 and 3-6.)

3-6. SBE Watchdog Timer

The watchdog timer detects when a program is no longer processing normally. This timer can be used to implement fail-safe operations for applications such as robot control where processor or software malfunction could cause irreparable damage.

Once the watchdog timer is activated, the program must periodically store a particular bit pattern into a specific memory location. If too much time elapses without storing this bit pattern, the watchdog times out causing the processor to reset.

The watchdog timer provides a timeout period of 0.1 seconds to 13 seconds depending on the jumper configuration of J25 and J26. Only install one jumper in these two areas. See Figure 3-6a.

Figure 3-6a: Watchdog Timeout Period

J Area	Timeout Period (In Seconds)*			
J25	A o B o 1	o o 2	o o 3	o o 4
	13.42	6.71	3.36	1.68
J26	A o B o 1	o o 2	o o 3	o o 4
	0.84	0.42	0.21	0.10

* Select only one of the eight options (i.e., install only one jumper.)

After a power up or reset, the watchdog timer is not active until an $AA byte is written to location $00F90701 for the first time. The Watchdog Timer Register is write-only (see Figure 3-6b below). Each time an $AA is moved to the watchdog timer register, the timer is reinitialized to count down toward 0. If the timer reaches 0, the processor is reset.

Figure 3-6b: Watchdog Timer Register (Address: $3F80F01 Write Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	1	0	1	0

<----------$AA---------->

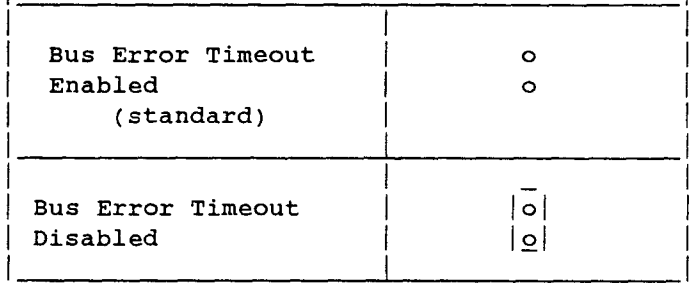
After being activated, there is no way to deactivate the timer except by resetting the MPU-28.

3-7. Bus Error Exception

In the standard configuration (i.e., no jumper installed), J30 is set to enable a bus error timeout on any access. A bus error exception occurs if a data access is not acknowledged within approximately 100 milliseconds. Since failure to acknowledge a data access is usually an error, this feature allows the system to obtain information about the cause of the error.

If a jumper is installed in J30, the processor will wait forever for the acknowledgment; the bus control and address lines will be "frozen" on the bus. This can be useful when debugging hardware problems.

Figure 3-7: J30 Bus Error Exception



3-8. Serial I/O Options

The MPU-28 provides two RS-232-C serial communication channels at 50-pin connector J6. The serial I/O channels are supported by an 8530 Serial Communications Controller. For information about the 8530, refer to the 8530 manual included in Section 9. An on-board oscillator provides a 3.6864MHz clock for internal timing and is suitable for standard serial baud rates.

High speed data transmission is possible using the on-board oscillator or by using external timing input to the 8530's bidirectional TXC pin.

The 8530 SCC can be programmed to generate six different interrupt vectors on Interrupt Level 5. Each of the three interrupt types for each channel - receive, transmit, and error condition - is associated with its own programmable interrupt vector number. This allows you to use high-speed interrupt-driven serial I/O for two channels on one interrupt level without resorting to polling techniques. See Section 6-5 for vectored-interrupt examples.

The MPU-28 includes jumper areas which control two major RS-232-C serial communication options:

1) The direction of each MPU-28 serial channel is jumper-selectable, allowing operation as either DCE or DTE. As DCE, the MPU-28 assumes the role of a central processing unit. As DTE, the MPU-28 assumes the role of a terminal or other peripheral device. Being able to use jumpers on the MPU-28 eliminates the need for a special cable when transmit & receive lines and handshake lines have to be exchanged.

2) The transmit clock may be received from a modem or it may be sent from the MPU-28 to a modem. (A transmit clock is necessary only for synchronous data transfer protocols.)

3-8-1. DCE/DTE Option

Jumper areas J16, J17, and J18 for Channel A and jumper areas J10, J11, and J12 for Channel B allow the serial channel to talk to either DTE (e.g., a terminal) or DCE (e.g., a modem or another computer). The following figures show the jumper configurations for each direction and also the signals associated with each jumper area.

Figure 3-8-1a: Jumpering For DCE/DTE Option

Channel A: J16, J17, J18
Channel B: J10, J11, J12

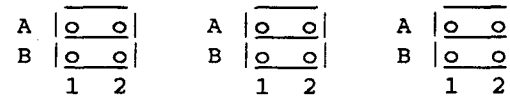
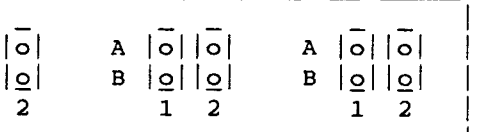
MPU-28 Configured As DCE To Talk To A Terminal (Or Other DTE)	MPU-28 Configured As DTE To Talk To A Modem (Or Other DCE)
	

Figure 3-8-1b: Signals Associated With J16-J18 & J10-J12

Channel A	Channel B	Associated Signal	
J16	J10	TXD/RXD	Transmit Data/Receive Data
J17	J11	RTS/CTS	Request To Send/Clear To Send
J18	J12	DTR/DCD	Data Terminal Ready/Data Carrier Detect

3-8-2. Transmit Clock Options

The MPU-28 allows the transmit clock to originate either from the MPU-28 or from a modem. When jumpered appropriately, the MPU-28 supplies the transmit clock on pin 24 of an RS-232-C connector or receives the transmit clock from pin 15 of an RS-232-C connector. Implementing this decision are jumper areas J14 and J15 for Channel A; J8 and J13 for Channel B. See the following figure.

Figure 3-8-2: Jumpering For Transmit Clock Source

Transmit Clock Source	J14: Channel A J8: Channel B	J15: Channel A J13: Channel B
MPU-28	o o	|o| |o|
Modem	|o| |o|	o o
Not Used	o o	o o

3-9. iSBX Connector

The MPU-28 has one iSBX connector, allowing you to choose from a wide selection of available 8- or 16-bit single- or double-wide iSBX modules. The industry-standard iSBX I/O expansion interface can be used to extend the parallel or serial capabilities of the board or to add features such as an extra serial port or time-of-day clock/calendar.

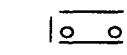
The iSBX connector has two select lines (MCS0 and MCS1) and, therefore, has two base addresses. A 16-bit iSBX module installed on iSBX connector J20 has base addresses of $03F80A00 and $03F80B00 for connector selects MCS0 and MCS1 respectively. Similarly, an 8-bit iSBX module has base addresses of $03F80A01 and $03F80B01.

The MPST* signal (Module Present) must be asserted in order for the MPU-28 to be able to receive interrupts from any Multimodule on the iSBX connector. Therefore, before installing a Multimodule on the MPU-28, make sure that MPST* is grounded on the module.

NOTE: Interrupt lines MINTR0 and MINTR1 are pulled high on the MPU-28. Since these are high-active lines, if an iSBX module is installed on J20, unused interrupt lines as well as MPST* must be grounded. Otherwise, the MPU-28 will receive a constant iSBX interrupt. (If an iSBX module is not installed, interrupts do not occur because MPST* is inactive.

If you are using an iSBX module that requires an additional address line from the microprocessor, a jumper must be installed at J44. When jumpered, J44 connects the A4 address line to pin 10 of J20, the iSBX connector. Most standard iSBX modules do not require this extra address line and thus a jumper need not be installed.

Figure 3-9: J44 iSBX Module Address Line

Address Line A4 Connected to iSBX Connector J20	
Address Line A4 not Connected to iSBX Connector J20	o o (standard)

3-10. SCSI Controller

The 53C90 SCSI controller provides a general interface between the SCSI bus and the 68020 processor. Designed to support the ANSI X3.131-1986 SCSI standard, the 53C90 enables the MPU-28 to perform high-speed data transfers to and from many printers and floppy, Winchester, and tape controllers. A complete 53C90 SCSI manual is included in Section 9.

The physical SCSI bus connection is implemented with single-ended signals through a 50-pin, dual-row SCSI bus connector. Able to support both initiator and target functions, the 53C90 can be programmed to operate in pseudo DMA or programmed I/O mode.

3-10-1. Pseudo-DMA Mode

The MPU-28 can be programmed to take advantage of the DMA capabilities of the 53C90 without the use of a DMA controller. To perform pseudo-DMA on the 53C90, the SCSI chip must first be programmed to supply data in DMA mode. When the data arrives, it is stored in the 53C90's FIFO and the DREQ bit in the control/status register (see Section 3-3) is set to 1.

The CPU can read one byte, from the 53C90 through the MPU-28's DMA/Data Acknowledge register at location $03F80D01.

When a byte is read from this register, on-board circuitry automatically issues a data acknowledge to the 53C90. The CPU can read the byte currently waiting in the FIFO buffer without further status checking or wasted time. When writing to this register the CPU can send one byte of data. This method of accessing the 53C90 greatly speeds up the data transfer process.

The layout of this byte-wide, read/write register follows:

Figure 3-10-1: Bits Of The DMA/Data Acknowledge Register
ADDRESS: $03F80D01

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

DB7-DB0 This byte of data is transferred between the SCSI's FIFO and the microprocessor while the 53C90 is operating in pseudo-DMA mode.

3-10-2. SCSI Identification Number

The microprocessor must assign a bus ID number to the SCSI controller because an ID for the SCSI is not initialized at power-up or reset. One way to assign an ID number is to put the ID in EPROM. Another is to program the microprocessor to read the switches in the control/status register and assign the SCSI controller an ID number based on the switch settings.

3-11. Parallel Interface/Timer (PI/T)

This section describes the features of the 68230 PI/T on the MPU-28. A complete 68230 manual is included in Section 9.

3-11-1. System Timer

The 68230 Parallel Interface/Timer contains a 24-bit synchronous down counter that can be used to measure time intervals or to generate periodic interrupts. The input frequency to the timer is provided by a 10MHz clock, not by the 12.5 or 20MHz processor clock.

3-11-2. Parallel Port

The 68230 Parallel Interface/Timer provides two 8-bit parallel I/O ports, i.e., ports A and B (PA0-PA7 and PB0-PB7); four handshake lines (H1-H4); and an 8-bit alternate function port C (PC0-PC7). The 16 parallel I/O lines are bidirectional; direction is controlled either through data direction registers or the handshake lines. Certain modes allow ports A and B to be concatenated to form a single 16-bit wide interface.

NOTE: Communication rates as high as 6.4 Mbits per second are possible when using two 68230 PI/Ts (e.g., in a communication link between two MPU-28 boards). When operating in 16-bit bidirectional mode 3, some external arbitration hardware may be required. "A Multiprocessor Interface" by Kim Eckert of Motorola in the November 1982 issue of IEEE Micro contains a circuit which exemplifies the kind of logic needed.

Some of the port C alternate function lines have specific assignments on the MPU-28. Other port C lines can be used for general purpose bidirectional I/O.

The following 68230 lines are brought directly to pins of I/O header J7:

Port A	PA0-PA7
Port B	PB0-PB7
Handshaking.	H1-H4
Port C	PC0-PC2, PC4

J7 pin assignments are given in Section 7 of this manual.

3-12. J22: Cache Disable

A major feature of the 68020 microprocessor is its on-chip cache memory which retains a copy of the most recently executed instructions. If program flow causes re-execution of an instruction in the cache, processing time is shortened since the instruction will not need to be fetched from program memory. Through careful use of the cache for software loops, program execution times can be reduced dramatically.

After a power up or reset, the cache memory feature is disabled. In the standard configuration of J22, the cache is enabled under program control using a MOVEC instruction to change the contents of the Cache Control Register (CACR). To prevent the cache memory feature from being enabled, install a jumper on J22. See the following figure.

Figure 3-12
J22: Preventing Enabling Of 68020 Memory Cache

Cache Is Enabled	Cache Is Not Enabled
o o (standard)	o o

3-13. Software Readable Switches

Eight read-only bits indicate the settings of eight switches located at the edge of the MPU-28 board. The eight switches are numbered 1-8 on the switch and are easily accessible with the MPU-28 installed in a Multibus. Setting a switch to the OPEN position causes the corresponding bit in the control/status register to be read as a 1. These eight switches make it convenient to assign board IDs or control program functions on multiple MPU-28 boards without using different EPROMs. Refer to Section 3-3 for bit assignment information.

3-14. Memory Management

On board the MPU-28 is memory mapping circuitry which enables the MPU-28 to control a multiuser/multitasking operating system. This circuitry performs two main functions:

(1) It translates memory addresses generated by the CPU into addresses in physical memory; and

(2) It provides complete system integrity, i.e., it protects memory from unauthorized accesses.

Some of the terminology surrounding memory management is complex so be sure to read this section before setting up memory maps. To begin, here are a few definitions of terms used to describe memory management.

Logical And Physical Addresses

Any memory address that the CPU wants translated by the memory mapping circuitry is called a **logical address**. The **physical address** is where data is physically stored.

Controlling Address Translation

The UE (user enable) and SE (supervisor enable) bits in the MMU control/status register control address translation when the CPU is in user or supervisor mode, respectively. If the appropriate bit is set to 1, translation is enabled; if it is set to 0, translation is disabled. See figures 3-14-2a and 3-14-2b.

Address Translations

A **memory map** translates one address into another within the CPU's local memory space. There are 32 separate memory maps available, with map 0 being reserved for use when in supervisor mode. Each map contains 512 entries, the contents of which are used to make up a physical address.

Page Size: 4K Or 64K

Memory mapping circuitry divides logical address space into 4 or 64Kbyte page sizes. Programs larger than 2Mbytes must use 64Kbyte pages. In supervisor mode, page size is determined by the S64 bit; in user mode it is determined by the U64 bit. If both supervisor and user modes are enabled in the same map, the page size for the different modes should be set the same (i.e., match the settings of the U64 and S64 bits). Since supervisor mode always uses memory map 0, the setting of S64 of the current memory map should agree with that of the supervisor page size of memory map 0. See figures 3-14-2a and 3-14-2b for more information.

CPU Access Modes

The 68020 CPU has two access modes:

User Mode The processor is allowed access to the user areas of memory while in user mode. If the UE bit in the MMU control/status register is set to 1, the address given by the processor is translated by the memory mapping circuitry, using the page size specified by the U64 bit.

If the UE bit is set to 0, the address is not translated and passes through unchanged. The MMU control/status register and memory map cannot be accessed while the processor is in user mode. Any attempt to access memory outside of the user area causes a bus error.

Supervisor Mode The processor is allowed to access all of memory while in supervisor mode. If the SE bit is set to 1, the memory mapping circuitry uses memory map 0 to translate the address. The page size is defined by the S64 bit in the MMU control/status register. If the SE bit in the MMU control/status register is set to 0, the address given by the processor is not translated and it passes through unchanged.

Global Memory Accesses

Memory mapping circuitry does not translate processor accesses of addresses in the global space. These accesses include interrupt acknowledges and coprocessor operations. User mode accesses above $08000000 are treated as global accesses. Global accesses are translated using memory map 0 and use the S64 bit to define the page size.

3-14-1. The Memory Map

The MPU-28's memory mapping circuitry can manage a myriad of tasks using up to 32 memory maps. Each map contains 512 entries that facilitate address translation from a virtual (or program-relative) address to an actual hardware physical address. A user can address up to 2Mbytes of memory by setting a given map to 4Kbyte pages via the MMU control/status register, or up to 32Mbytes with 64Kbyte pages.

Depending on the page size of the selected memory map (either 4 or 64Kbyte pages), certain bits of the virtual address will index into one of the 512 entries and use that entry's data to form the physical address. Setting up a memory map, therefore, entails setting up the required number of 512 memory map entries. The entry also contains access permissions (read only, write only, read/write, and no access).

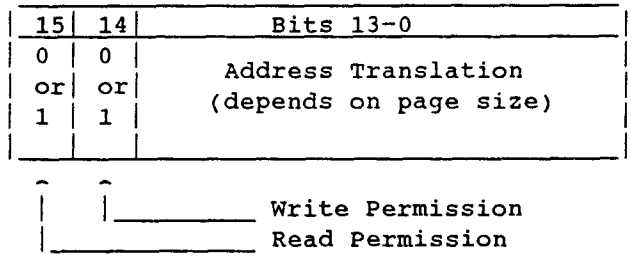


Figure 3-14-1: Typical Memory Map Entry

Note that memory map 0 is always used when in supervisor mode. The exception is for direct addressing of the maps (i.e., address bit 27=1) when direct on-board addressing is allowed. When the processor is executing in supervisor mode, the MMU control/status register need not be changed from a previous memory map since the supervisor's automatic use of map 0 is controlled via hardware. Any instruction executed in user mode will revert to the previously selected map.

3-14-2. MMU Control/Status Register

Bits of the MMU control/status register can only be accessed while the processor is in supervisor mode and cannot be accessed in user mode without causing a bus error.

The map number (0-31) must be loaded into the MMU control/status register before any of that map's 512 entries can be accessed. The required map number (0-31) is loading into bits 0-4. The map is then enabled for either supervisor and/or user accesses, and then enabled for the correct page size. Normal operation would enable one mode, but not both in a given map.

The bits of the upper byte of the MMU control/status register are as follows:

Figure 3-14-2a: Upper Byte Of The MMU Control/Status Register
ADDRESS: $08000000

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
unused	unused	unused	unused	unused	unused	S64	SE

unused — Bits 10-15 are unused.

S64 — Supervisor accesses are mapped in 4Kbyte pages when this bit is set to 0. Set to 1, supervisor accesses are mapped in 64Kbyte pages. This bit is ignored if the SE bit is set to 0. When configuring a supervisor or user memory map, this bit should be set to reflect the page size (4 or 64Kbyte) in which the map will be used.

SE — This bit enables memory mapping while the processor is in supervisor mode. If this bit is 0, no mapping takes place in supervisor mode and the address is not modified before use. If this bit is 1, mapping is performed using memory map 0 regardless of the value of the UAx bits in the MMU control/status register.

The bits of the lower byte of the MMU control/status register are as follows:

Figure 3-14-2b: Lower Byte Of The MMU Control/Status Register
ADDRESS: $08000001

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
unused	U64	UE	UA4	UA3	UA2	UA1	UA0

unused — Bit 7 is unused.

U64 — If this bit is set to 0, user accesses are mapped in 4Kbyte pages. Set to 1, user accesses are mapped in 64Kbyte pages. This bit is ignored if the UE bit is set to 0.

UE — This bit enables memory mapping while the processor is in user mode. If this bit is 0, no mapping takes place in user mode and the address is not modified before use. If this bit is a 1, mapping is performed using the map selected by the UAx bits in the MMU control/status register.

If both supervisor and user modes are enabled in the same map, predictable results can be obtained by matching the settings of the U64 and S64 bits. Remember; however, that supervisor mode always defaults to using memory map 0.

UA4-UA0 — These five bits select which memory map (0 through 31) is used to perform address translations in user mode. When supervisor mapping is performed, map 0 is used regardless of the setting of the UAx bits. These bits also define which map is accessed when the map is being configured.

3-14-3. How Logical Addresses Are Converted To Physical Addresses

There are two types of conversions: one for Multibus addresses (which will not be discussed here -- see Section 4-6) using the translation maps, and one for CPU local addresses using the memory maps. Converting logical addresses uses the memory maps.

When the CPU performs a local memory access, some of its address lines are delivered to the memory mapping circuitry on the MPU-28 board, while other address lines are sent straight through to MPU-28 memory. Depending on the selected page size, different bits will be passed through or translated.

In 64Kbyte operation, the 16 low-order bits and A25 are passed straight through to MPU-28 memory. The next nine address bits (A16-A24) are the index into the selected map's entries, while address bits A29-A31 are ignored. In addition, A26, A27, and A28 are passed straight through to Multibus I/O, MMU control/status register, and MMU maps, respectively. The memory map entry, which the nine bits select, contains the new values to be used for the nine address bits A16-A24.

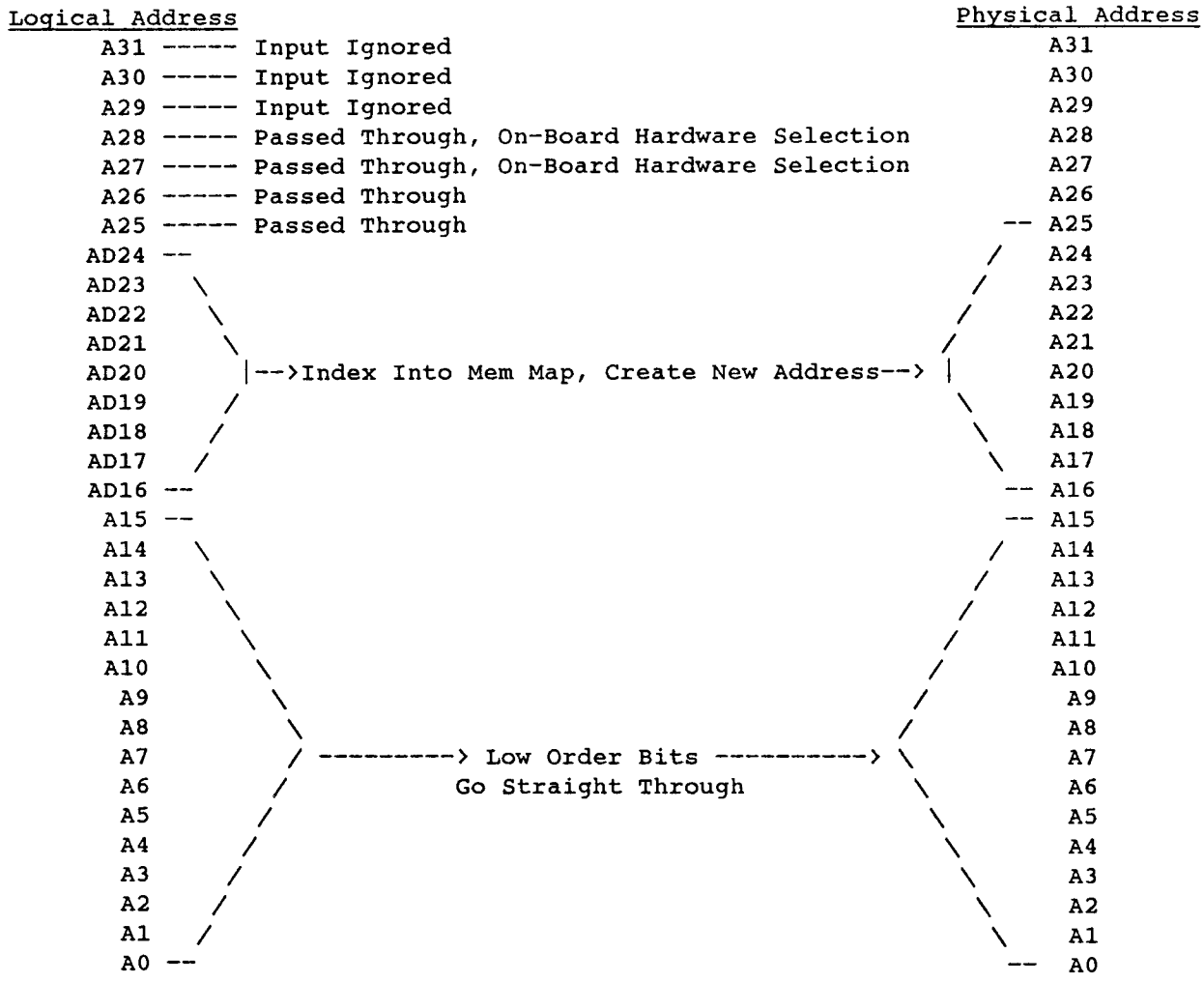


Figure 3-14-3a: 64Kbyte Page Size Address Translation

In 4Kbyte operation, the 12 low-order address bits and AD25 are passed straight through. The nine address bits (A12-A20) are the index into the selected map's entries, while address bits A21-A24 and A29-A31 are ignored. The memory map entry, which the nine bits select, contains the new values to be used for the 14 address bits A12-A25.

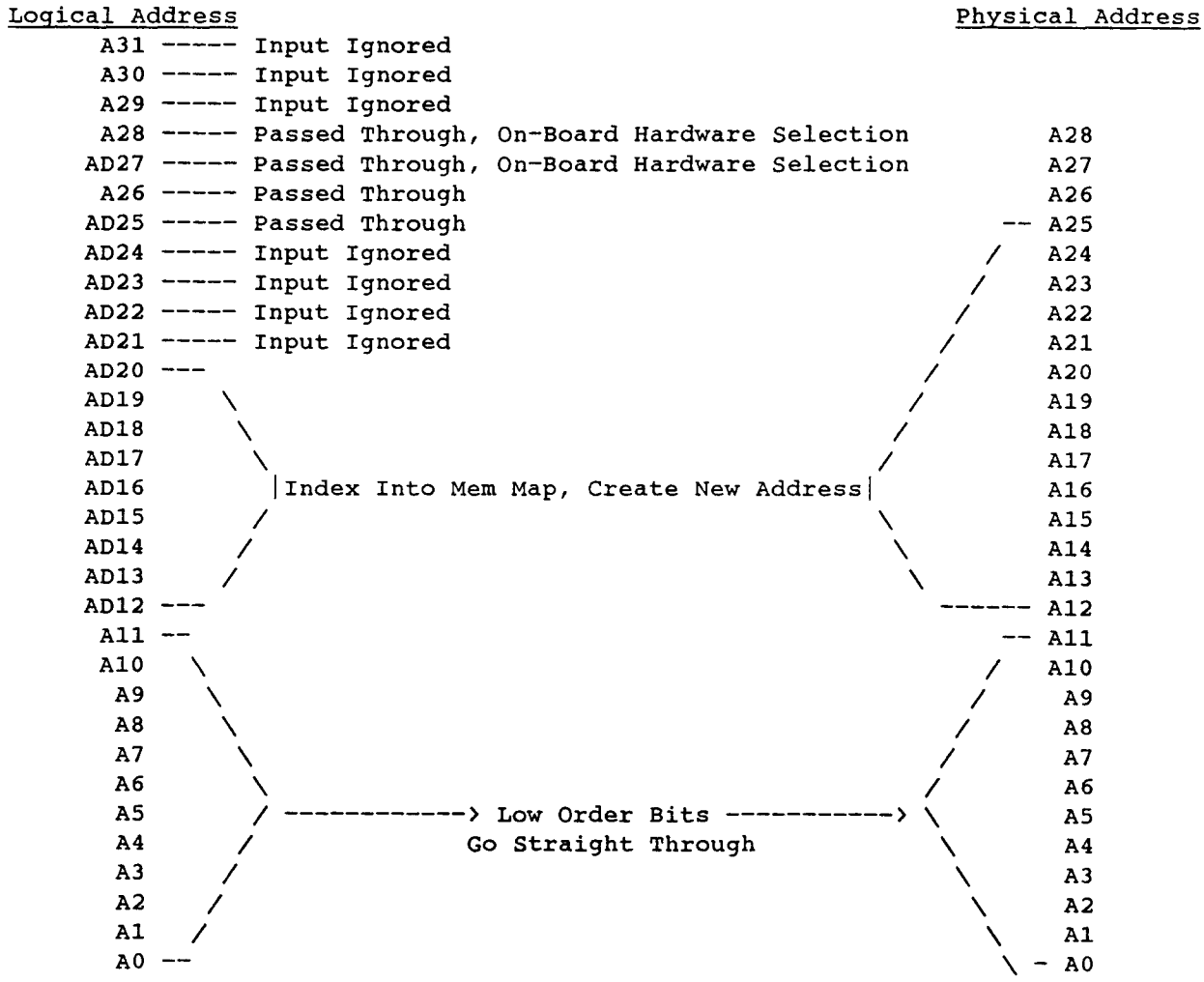


Figure 3-14-3b: 4Kbyte Page Size Address Translation

3-14-4. Address Translation

To configure a given memory map, the map must first be selected by loading the map number into the MMU control/status register. Then, the correct address translation must be calculated and stored into one of the 512 possible memory map entries.

The algorithm for calculating the value to be entered into a memory map entry is as follows:

```
4K:  Write [(PA/SZ) + AP] to location [MA + ((LA & MASK4) <<4)]
64K: Write [(PA/SZ) + AP] to location [MA + (LA & MASK64)]

     Where:  PA = Physical Address
             SZ = $1000
             AP = Access Permissions
                  0000 = no access allowed
                  4000 = write only
                  8000 = read only
                  C000 = read/write
             LA = Logical Address
             MA = $18000000
          MASK4 = 0x01FF000
         MASK64 = 0x1FF0000
```

Let's say, for example, you want the MPU-28 to translate a logical address of $1000 to a physical address of $20000 and want to be able to read and write to that address using 4Kbyte page sizes. The following steps update the memory map currently pointed to by the MMU control/status register:

1. Divide the physical address $20000 by $1000 (4096).
   $20000/$1000 = $20

2. Add the code for read/write permissions ($C000) to the sum of Step 1.
   $20 + $C000 = $C020

3. Obtain the significant portion of the logical address and, since this is a 4Kbyte page size, shift left four bits.
   $1000 & 0x1FF000 = $1000
   shift left four bits = $10000

4. Calculate the physical address of the map entry.
   $18000000 + $10000 = $18010000

5. Write the value calculated in Step 2 to the address location of Step 4.
   Write $C020 into $18010000

------ NOTES ------

4. THE MPU-28 AND THE MULTIBUS

A Multibus **master** is a microprocessor card such as SBE's MPU-28. A master can also be an I/O card that can take control of the Multibus in order to transfer data to/from memory, such as SBE's COM-8 Serial Communication Board.

A **single-master** Multibus system is one in which only one master is installed on the Multibus. In such a system, this master retains control of the bus at all times.

A **multimaster** Multibus system is one in which more than one board can function as master of the Multibus, for example, a system with an MPU-28 and a COM-2. In such a system, the masters must share control of the bus.

In a multimaster system, only one master drives the signals BCLK (Bus Clock) and CCLK (Common Clock). This master is referred to as the **primary master**. The other masters in the system, **secondary masters**, must be set up to receive the BCLK and CCLK signals from the primary master.

NOTE: The primary master of the system should be placed in slot 1 of most Multibus card cages (the slot furthest away from the terminating resistors) to avoid possible excessive noise or ringing.

4-1. Serial Vs. Parallel Priority Scheme

In a multimaster system, the masters must arbitrate for use of the Multibus. To do this, a bus priority level is associated with each master. The priority level is used to determine which master will have access to the Multibus at any given moment.

There are two schemes for assigning priority levels to masters: serial priority and parallel priority. If the system has only two or three masters, the serial priority scheme can be used. The parallel priority scheme must be used when there are more than three masters in a system or when you wish to assign priority independent of each board's order in the Multibus backplane.

Serial Priority Scheme. In a serial priority scheme, the priority of each master is determined by its order in the Multibus card cage: the highest priority master resides in the first slot, the second-highest priority master resides in the next slot, etc.

With serial priority, each board must be jumpered to generate the BPRO* signal. The boards are daisy-chained together with the BPRO* (Bus Priority Out) signal of each board feeding the BPRN* (Bus Priority In) signal of the next board. The BPRN* signal of the highest priority master must be connected to ground. If a slot on the backplane is to be left empty, the BPRO* signal must be brought over the empty slot to the BPRN* signal of the next slot. This is usually done by wire-wrapping the backplane.

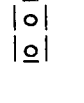
Parallel Priority Scheme. In a parallel priority scheme, the priority of each Multibus slot is determined by a parallel priority encoder on the Multibus backplane. The BREQ* (Bus Request) and BPRN* (Bus Priority In) signals of the slots are individually wired to the parallel priority encoder. This allows the priority of the slots to be assigned in any order. Any of the slots can be unused if desired and even slots that are wired up need not be filled.

With parallel priority, the priority of each master is determined by the slot into which it is plugged. Each board must be jumpered <u>not</u> to drive the BPRO* signal.

The highest-priority master in a multimaster system is not necessarily the primary master; therefore, the board receiving highest priority for Multibus accesses does not necessarily drive the BCLK and CCLK signals.

4-2. Bus Priority Jumpering

Figure 4-2: Bus Priority Jumpering

System	J35: Multibus BPRO* Signal	J36: Multibus BPRN* Signal	J39: Multibus CBRQ* Signal
Single-Master System	J35 jumpering does not matter.	 Standard: BPRN* Received From Multibus o o +BPRN* Asserted	 Standard: MPU-28 releases Multibus only if requested to do so.
Multimaster System: Serial Priority Scheme	 Standard: BPRO* Connected	 Standard: BPRN* is received from Multibus (for not highest priority master). o o +BPRN* Asserted On Highest Prior. Master	++Depends On Use
Multimaster System: Parallel Priority Scheme	o o BPRO* Not Connected	 BPRN* Received From Multibus	++Depends On Use

\+ This is necessary only if BPRN* is not grounded on the backplane.
++ Section 4-3 explains how to jumper J39 for your application.

4-3. Release Of Multibus After Access

Jumper area J39 determines the action taken by the MPU-28 following each Multibus access.

If no jumper is installed in J39, the MPU-28 gives up mastership of the bus following each Multibus access. This allows other masters more use of the Multibus, but results in slower bus access for the MPU-28 (since bus re-arbitration is required for each MPU-28 access).

If a jumper is installed in J39, bus arbitration is required only if another master needs to use the bus; therefore, the MPU-28 remains bus master between accesses, without going through the process of bus arbitration for each access. This option is recommended where practical. The Multibus signal CBRQ* (Common Bus Request) allows the current master to sense whether another master needs the bus.

In the standard configuration (a jumper installed at J39), the MPU-28 master will not release the bus to lower-priority masters unless they request the bus. To prevent bus "hogging" by high-priority masters, they should access the bus only for short periods of time (e.g., for short bursts of DMA). For example, an MPU-28 set up as the highest-priority master can fully saturate the Multibus while executing a program out of Multibus memory (i.e., while fetching instructions over the Multibus). Lower-priority masters are prevented from gaining access to the bus as long as the program is running. Therefore, a master that will be executing code from memory on the Multibus should be assigned a low priority.

Figure 4-3: Jumpering J39 For Release Of Multibus

o o	|o| |o|
MPU-28 releases Multibus after each access.	MPU-28 releases Multibus only if requested to do so. (standard)

4-4. Multibus Clock Option Jumpering (BCLK & CCLK)

The MPU-28 can be jumpered either as a primary Multibus master, which supplies the BCLK (Bus Clock) and CCLK (Common Clock) signals to the Multibus, or as a secondary Multibus master, which receives those signals from the Multibus. J34 and J37 control the direction of BCLK and CCLK, respectively.

In its standard configuration, the MPU-28 is jumpered as a primary master.

NOTE: Be sure that only one module in your system - the primary master - supplies the BCLK and CCLK signals. The system may behave erratically if more than one board supplies BCLK and CCLK signals or if no board drives these clocks.

Figure 4-4: Clock Option Jumpering (BCLK & CCLK)

Condition	J34 (BCLK)	J37 (CCLK)
MPU-28 as a primary master (or in a stand-alone system): set up to supply BCLK & CCLK to the Multibus (standard)	o o	o o
MPU-28 as a secondary master: set up to receive BCLK & CCLK from the Multibus	o o	o o

4-5. Multibus I/O Address Space

The Multibus specifications call for two independent address spaces - one for memory (which is discussed in the next section) and another for I/O. The Multibus I/O address space is only 64Kbytes wide, yet, like the memory address space, it supports both 8-bit and 16-bit transfers.

The MPU-28 accesses the Multibus I/O space through two 64Kbyte windows. Although both address ranges access the same 64Kbyte block, memory accesses in the address range $03F90000 through $03F9FFFF force byte swapping, whereas memory accesses in the address range $07F90000 through $07F9FFFF handle data without byte swapping. Byte swapping is discussed in sections 4-6 and 4-7.

The MPU-28 is able to access its own mailbox address through the Multibus I/O 64Kbyte window. Therefore, it can send a mailbox interrupt to itself. (See Section 3-5.)

The SBE Mailbox implements the only functions on the MPU-28 that are utilized by other Multibus masters using the Multibus I/O address space.

4-6. MPU-28 Accesses To Multibus Memory

The Multibus memory space is 16Mbytes wide, requiring 24 bits to address an individual byte. The MPU-28 has two 16Mbyte address windows for accessing Multibus memory as shown in the following figure.

Figure 4-6a
Byte Swapping Options For CPU Accesses To Multibus Memory

Address Range	Swapping?	Window Size	J28
$01000000-$01FFFFFF	Bytes Swapped	16Mbytes	o o o 1 2 3
$05000000-$05FFFFFF	Bytes Not Swapped	16Mbytes	o o o 1 2 3

The 68020 CPU stores the two bytes within a 16-bit word in the opposite order from the Multibus standard. To compensate for this, the MPU-28 can be set to swap bytes automatically when doing Multibus memory accesses. With this option, the value of the low-order address bit is converted so that bytes as well as words are accessed correctly.

As shown in Figure 4-6a, the program running on the MPU-28 can control byte swapping by using the appropriate address range. Memory accesses in the $01000000 through $01FFFFFF address range will use byte swapping; memory accesses in the $05000000 through $05FFFFFF range will not.

For example, a program would access Multibus memory address $FCB123 without byte swapping by using the corresponding address in the range $05000000 through $05FFFFFF: $05FCB123. (The same Multibus memory would be accessed with byte swapping using address $01FCB123.)

Jumper area J28 allows setting up the MPU-28 to control byte swapping via the BS bit (bit 4 at $03F80901), in the control/status register, instead of alternate address windows. If a jumper is installed connecting pins 2 and 3 of J28, the status of BS controls byte swapping. Clearing BS to 0 causes byte swapping on all MPU-28 accesses to Multibus memory. A 1 in BS causes unswapped accesses. BS is cleared to 0 on power up or reset.

Figure 4-6b
J28: Selection Of Byte Swap Control Methods

Byte Swapping Controlled by:	J28
Address of Access to Multibus Memory (standard)	 1 2 3
BS Bit in the System Control Register	 1 2 3

When To Use Byte Swapping. Use byte swapping if the MPU-28 is sharing access to the same data as a board that does byte swapping (e.g., an SBE M68K10) or a board that stores its data in byte (not word) mode (e.g., a board with an 8-bit Intel processor).

4-7. Multibus Accesses To The MPU-28: Shared Memory

When the MPU-28 operates in a multimaster system, other masters may access the MPU-28's on-board dynamic RAM through the Multibus. The amount of RAM on the MPU-28 is 1, 2, 4, or 8Mbytes depending on the size of the RAM chips and whether four or eight SIP modules are installed.

All or part of the MPU-28's dynamic memory may be mapped into the Multibus memory address space. This means that other masters on the Multibus may read from and write to the MPU-28's on-board RAM. All Multibus accesses to on-board memory use byte swapping. When another board on the Multibus is accessing memory on the MPU-28, local memory accesses by the local processor are disabled.

4-7-1. Overview Of Multibus Accesses To MPU-28 Shared Memory

Several features of the MPU-28 provide flexibility and control when handling Multibus accesses of MPU-28 RAM. This section presents an overview of the process; later sections deal with the features and functions in detail.

An access to MPU-28 RAM passes through two stages: address recognition followed by address translation.

Address recognition occurs when another device makes a Multibus memory access which the MPU-28 determines is within the address window of its RAM. The starting address and size of this window is controlled by jumper area J42. The size can be as small as 64Kbytes or as large as 16Mbytes. The starting address can be set in the 16Mbyte Multibus address space on a boundary that is a multiple of that window size.

Once the MPU-28 has recognized an access to its RAM, it translates the access address into an on-board memory address. The upper 8 bits of the access address are translated into the upper 8 bits of the on-board address using a memory map set up and maintained by the MPU-28 program. The translated 8 bits are combined with the lower 16 bits of the original access address forming a full 24-bit address to an on-board RAM location.

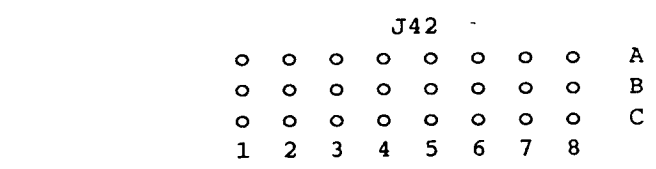
One translation map is sufficient for mapping the entire Multibus memory space to some or all of MPU-28 RAM. However, to facilitate changing translation schemes during operation, the MPU-28 has four complete translation maps. The MPU-28 program can instantaneously switch which map is currently active through use of the MS0 and MS1 bits in control/status register Byte 1. (See Section 3-3.)

4-7-2. Address Recognition: Multibus Address Of Shared Memory

Jumper area J42 is used to select the starting address in Multibus address space and the window size of MPU-28 shared memory.

The eight columns of jumper J42 correspond to the eight upper bits of the Multibus access starting address. A bit value of 0 is set by installing a jumper over A and B; installing no jumper establishes a value of 1 for the bit. Figure 4-7-2a shows how to jumper J42 for the starting address of the shared memory.

Figure 4-7-2a
J42: Specifying Multibus Address Space Of Shared Memory

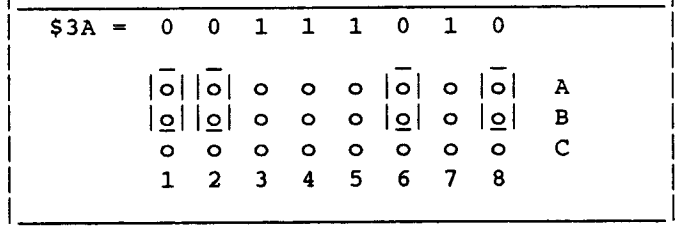


		Jumper Configuration		
Multibus Line*	Corresponding J42 Column	Bit Value of 1 on Address Line	Bit Value of 0 on Address Line	Bit Value on Address line is Ignored
ADR17	1	No Jumper	A to B	B to C
ADR16	2	No Jumper	A to B	B to C
ADR15	3	No Jumper	A to B	B to C
ADR14	4	No Jumper	A to B	B to C
ADR13	5	No Jumper	A to B	B to C
ADR12	6	No Jumper	A to B	B to C
ADR11	7	No Jumper	A to B	B to C
ADR10	8	No Jumper	A to B	B to C

* The Multibus specification uses hexadecimal numbering of the address lines. Hence, ADR17 corresponds to A23, etc.

Let's say for example, you want other Multibus masters to access shared memory between $3A0000 through $3AFFFF. In this case, you jumper J42, as shown below, to represent the upper eight bits of the starting address (i.e., $3A in hexadecimal; 0011 1010 in binary). Remember that 0 values have jumpers; 1 values have no jumpers.

Figure 4-7-2b
J42 Jumpering Example: $3A0000-$3AFFFF

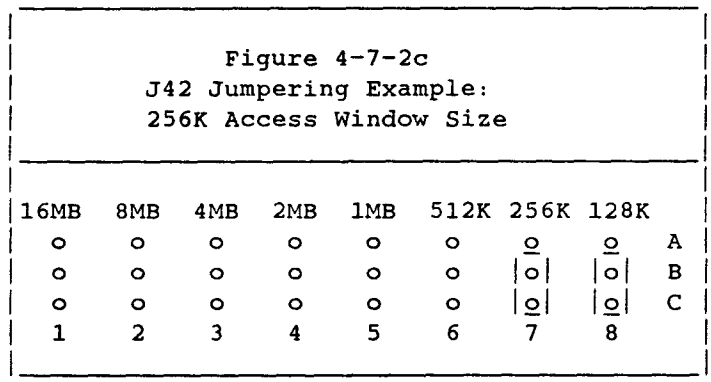


The eight columns of jumper J42 also correspond to the window size of RAM accesses. Possible window sizes are 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, or 16Mbytes. If your RAM access window size is 64Kbytes, no jumper is installed on rows B and C of J42. If you want to access more than 64Kbytes of shared memory, for instance 256K, positions 7 and 8 of J42 are jumpered B to C. Because you have the remaining six positions for setting the starting address, you are able to choose any 256K boundary.

Similarly, if your access window size is 512Kbytes, five positions at J42 are available for setting the address on any 512K boundary.

The following figure shows what RAM access sizes are possible using J42.

Figure 4-7-2c
J42 Jumpering Example:
256K Access Window Size



CAUTION: **The MPU-28 cannot use the Multibus to access its own on-board memory. If the MPU-28 tries to access this block of Multibus memory, the board will lock up in arbitration for the Multibus and require a reset.**

4-7-3. Address Translation: Multibus Accesses To Shared Memory

The MPU-28 can have up to 8Mbytes of dynamic RAM on-board. This on-board memory space always begins at address $00000000 as viewed by the MPU-28 processor and circuitry.

The on-board address space, $00000000-$00FFFFFF, is segmented into 256 64Kbyte blocks. Each block is uniquely identified by the 8 bits, A23-A16, of its on-board address. Thus, the first 64K block, from $00000000 to $0000FFFF, is identified by $00; the second block, from $00010000 to $0001FFFF, is identified by $01; etc.. The last 64Kbyte block for a 8Mbyte MPU-28 (from $007F0000 to $007FFFFF) is identified by $7F.

If other Multibus masters will access on-board memory, at least one of the four available translation maps will have to be initialized with block identifiers. Each map consists of 256 8-bit entries. The entry to be used in a translation is selected by the upper 8 bits of the access address on the Multibus. In this manner, the upper 8 bits of the Multibus address are used to select the corresponding entry in the active translation map; the value for that entry is a programmed 8-bit identifier of a 64K block of on-board memory. The specific location within the 64K block is determined by the low-order 16 bits of the access address from the Multibus.

There are four independent translation maps on the MPU-28. The program controls which map will be used during an off-board access to shared memory via bits MS0 and MS1 in the control/status register byte at $03F80901.

Figure 4-7-3a: MS0/MS1 Selection Of Active Translation Map

Active Map	MS1	MS0
0	0	0
1	0	1
2	1	0
3	1	1

Following a power-up or reset, both MS0 and MS1 are initialized to 0, thereby selecting translation map 0. Many applications will not need more than this one map.

Map entries are **not** set to particular values at power up. Therefore, it is customary to initialize at least map 0 following a power-up or reset. Until the initialization of the active map is complete, other Multibus devices should not attempt to access the MPU-28's RAM. Completed initialization can be signaled to other Multibus masters via the READY bit in the Mailbox Status Register.

Initialization of a translation map is simply a matter of storing the desired pattern of on-board memory block identifiers into the MPU-28's memory map. Each entry byte is written at an odd byte address and, therefore, the 256-byte entries for one map require a $200 address space.

NOTE: Map entries are accessed as words at even addresses; however, only the lower byte is valid.

The addresses to use for the four translation maps are shown in the following figure. Note that the values of MS0 and MS1 are not involved in initializing or altering translation map entries: individual maps are selected on the basis of address.

Figure 4-7-3b
Translation Map Setup Addresses

Map	Address Range
0	$03F80000 - $03F801FF
1	$03F80200 - $03F803FF
2	$03F80400 - $03F805FF
3	$03F80600 - $03F807FF

Note - Entry bytes must be stored at odd byte addresses.

In many applications, it isn't necessary to initialize all 256 entries of an active map. Except when J42 is jumpered so all 8 high-order bits of the access address are recognized, some map entries are unnecessary -- not all Multibus addresses will get past the address recognition stage. In the example in the preceding section where J42 was jumpered to recognize only one 64K address range from $3A0000 through $3AFFFF, only the single map entry associated with $3A (i.e., the entry at $03F80074 for translation map 0) will ever be used.

NOTE: If an access from the Multibus is mapped to nonexistent MPU-28 memory, the data from a read cannot be used and data from a write will be lost.

4-8. Reset Of The MPU-28

There are two types of reset on the MPU-28 board. "Full reset" means that all I/O devices and the CPU receive a reset pulse. A "partial reset", which occurs when the CPU executes a reset instruction, means that all I/O devices receive a reset pulse but the CPU does not. This section first outlines the five ways to reset the MPU-28, then shows the various options concerning the INIT line.

There are five ways to reset the MPU-28:

1) Power up the MPU-28. This is a full reset.
2) Connect and then disconnect pins 5 and 6 of connector J4. (This is generally done by wiring a reset switch to these pins.) This is a full reset. See the figures in Section 7-6.
3) Use the INIT line from the Multibus. This is a full reset.
4) From the Multibus to the MPU-28 mailbox address, write data with bits 0 and 1 both set to 0. This is a full reset.
5) Use a 68020 reset instruction. This is a partial reset.

4-8-1. Use Of The INIT Line

The Multibus INIT signal is used to reset the boards in a Multibus system. The INIT signal may be supplied by one of the boards on the Multibus or by circuitry connected to the Multibus backplane. In either case, the INIT signal must be driven by only one source and the other boards on the Multibus must receive or ignore the INIT signal.

The MPU-28 can be jumpered to drive, receive, or ignore the INIT signal. When the MPU-28 is jumpered to drive the Multibus INIT signal, INIT is asserted when the reset switch is used. As shown in Figure 4-8-1, J19 controls whether INIT is also asserted when the MPU-28 executes a reset instruction.

When INIT is to be received by the MPU-28, J33 must be jumpered **not** to supply INIT to the Multibus and J19 must be set to receive the INIT signal from the Multibus. See the following figure. When the MPU-28 is set up to receive the INIT signal, a full reset of the MPU-28 occurs when INIT is active. Neither an MPU-28 reset instruction nor an MPU-28 reset switch affects the Multibus.

Figure 4-8-1: Reset Option Jumpering

Function	J19	J38	J33
Receive INIT	A |o| o B |o| o C o o 1 2	|o| |o|	o o
Drive INIT: Only Reset Switch Does Reset	A |o o| B o |o| C o |o| 1 2	o o	|o| |o|
Drive INIT: Reset Instruction As Well As Reset Switch Does Reset (Standard)	A |o o| B |o| o C |o| o 1 2	o o	|o| |o|
Ignore INIT	A |o o| B o o C o o 1 2	o o	o o

5. FLOATING-POINT COPROCESSOR*

The 68881 or 68882 floating-point coprocessor is a single chip that can be installed on the MPU-28 to overlap floating-point calculations with 68020 instruction execution. Following is an overview of the 68881 and 68882 from the Motorola MC68881/MC68882 Floating-Point Coprocessor User's Manual. This information was adapted with permission. If you require further information, the 68881/68882 manual can be ordered from Motorola Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721.

The 68881 and 68882 floating-point coprocessors (FPCP) both fully implement the IEEE Standard for Binary Floating-Point Arithmetic (ANSI-IEEE Std 754-1985). The 68881 and 68882 provide a logical extension to the 68020's integer data processing capabilities. They do this by providing a very high performance floating-point arithmetic unit and a set of floating-point data registers that are utilized in a manner that is analogous to the earlier members of the 68000 Family which support all of the addressing modes of the 68020.

The major features of the 68881 are:

- Eight general purpose floating-point data registers, each supporting a full 80-bit extended precision real data format (a 64-bit mantissa plus a sign bit, and a 15-bit signed exponent).
- A 67-bit arithmetic unit to allow very fast calculations, with intermediate precision greater than the extended precision format.
- A 67-bit barrel shifter for high-speed shifting operations (for normalizing etc.).
- Forty-six instructions, including 35 arithmetic operations.
- Full conformation to the IEEE P754 standard, including all requirements and suggestions.
- Support of functions not defined by the IEEE standard, including a full set of trigonometric and transcendental functions.
- Seven data types: byte, word and long integers; single, double, and extended precision real numbers; and packed binary coded decimal string real numbers.
- Twenty-two constants available in the on-chip ROM, including pi, e, and powers of 10.
- Virtual memory/machine operations.
- Efficient mechanisms for procedure calls, context switches, and interrupt handling.
- Fully concurrent instruction execution with the main processor.

In addition to these features, the 68882 provides:

- Concurrent execution of multiple floating-point instructions.

- Special purpose hardware for high-speed conversion of binary real memory operands to/from the internal extended format.

*** Floating-point coprocessor information supplied courtesy of Motorola, Inc.**

5-1. The Coprocessor Concept

The FPCP utilizes the 68020 coprocessor interface to provide a logical extension of the 68020 registers and instruction set in a manner which is transparent to the programmer. The programmer perceives the 68020/FPCP execution model as if both devices were implemented on one chip.

A fundamental goal of the coprocessor interface is to provide the programmer with an execution model based upon sequential instruction execution by the 68020 and the FPCP. For optimum performance, however, the coprocessor interface allows concurrent operations in the FPCP with respect to the 68020 whenever possible. In order to simplify the programmer's model, the coprocessor interface is designed to emulate, as closely as possible, non-concurrent operation between the 68020 and the FPCP.

The FPCP is a non-DMA type coprocessor which uses a subset of the general purpose coprocessor interface supported by the 68020. Features of the interface implemented in the FPCP are as follows:

- The 68020 and FPCP communicate via standard 68000 bus cycles.
- 68020 and FPCP communications are not dependent upon the instruction sets or internal details of the individual devices (e.g., instruction pipes or caches, addressing modes).
- FPCP instructions utilize all addressing modes provided by the 68020; all effective addresses are calculated by the 68020 at the request of the coprocessor.
- All data transfers are performed by the 68020 at the request of the FPCP; thus memory management, bus errors, address errors, and bus arbitration function as if the FPCP instructions are executed by the main processor.
- Overlapped (concurrent) instruction execution enhances throughput while maintaining the programmer's model of sequential instruction execution.
- Coprocessor detection of exceptions, which require a trap to be taken, are serviced by the 68020 at the request of the FPCP; thus exception processing functions as if the FPCP instructions were executed by the main processor.
- Support of virtual memory/virtual machine systems is provided via the FSAVE and FRESTORE instructions.
- Systems may use software emulation of the FPCP without reassembling or relinking user software.

5-2. Hardware Overview

The architecture of the FPCP appears to the user as a logical extension of the 68000 Family architecture. Because of the coupling of the coprocessor interface, the 68020 programmer can view the FPCP registers as though the registers are resident in the 68020. Thus, the 68020/FPCP device pair appears to be one processor that supports seven floating-point and integer data types, with eight integer data registers, eight address registers, and eight floating-point data registers.

The FPCP programming model consists of the following:

* Eight 80-bit floating-point data registers (FP0-FP7). These registers are analogous to the integer data registers (D0-D7) and are completely general purpose (i.e., any instruction may use any register).

* A 32-bit control register that contains enable bits for each class of exception trap, and mode bits to set the user-selectable rounding and precision modes.

* A 32-bit status register that contains floating-point condition codes, quotient bits, and exception status information.

* A 32-bit instruction address register that contains the main processor memory address of the last floating-point instruction that was executed. This address is used in exception handling to locate the instruction that caused the exception.

5-3. Bus Interface Unit

All communications between the 68020 and the FPCP occur via standard 68000 Family bus transfers. The FPCP contains a number of coprocessor interface registers (CIRs) which are addressed in the same manner as memory by the main processor.

When the 68020 detects a typical FPCP instruction, the 68020 writes the instruction to the memory-mapped command CIR, and reads the response CIR. In this response, the bus interface unit (BIU) encodes requests for any additional action required of the 68020 on behalf of the FPCP. For example, the response may request that the 68020 fetch an operand from the evaluated effective address and transfer the operand to the operand CIR. Once the 68020 fulfills the coprocessor request(s), the 68020 is free to fetch and execute subsequent instructions.

A key concern in a coprocessor interface that allows concurrent instruction execution is synchronization during main processor and coprocessor communication. If a subsequent instruction is written to the FPCP before it has completed execution of the previous instruction, the response instructs the 68020 to wait. Thus, the choice of concurrent or nonconcurrent instruction execution is determined on an instruction-by-instruction basis by the coprocessor.

5-4. Coprocessor Interface

The 68000 Family coprocessor interface is an integral part of the FPCP and 68020 design, with the interface tasks shared between the two. Tasks are partitioned such that the 68020 does not have to decode coprocessor instructions, and the FPCP does not have to duplicate main processor functions such as effective address evaluation.

This partitioning provides an orthogonal extension of the instruction set by permitting FPCP instruction to utilize all 68020 addressing modes and to generate execution time exception traps. Thus, from the programmer's view, the CPU and coprocessor appear to be integrated onto a single chip. While the execution of the great majority of FPCP instructions may be overlapped with the execution of 68020 instruction, concurrency is completely transparent to the programmer. The 68020 single-step and program flow (trace) modes are fully supported by the FPCP/68020 coprocessor interface.

While the 68000 Family coprocessor interface permits coprocessors to be bus masters, the FPCP is never a bus master. The FPCP requests that the 68020 fetch all operands and store all results. In this manner, the 68020 32-bit data bus provides high speed transfer of floating-point operands and results while simplifying the design of the FPCP.

Since the coprocessor interface consists solely of bus cycles (to and from the CPU space) and the FPCP never functions as a bus master, the coprocessor can be placed on either the logical or physical side of the system memory management unit in a 68020 system.

The virtual machine architecture of the 68020 is supported by the coprocessor interface and the FPCP through the FSAVE and FRESTORE instructions. If the 68020 detects a page fault and/or a task timeout, the 68020 can force the FPCP to stop whatever operation is in process at any time (even in the middle of the execution of an instruction) and save the FPCP internal state in memory. During the execution of a floating-point instruction, the FPCP can stop at predetermined points as well as at the completion of the instruction.

The size of the saved internal state of the FPCP is dependent upon what it is doing at the time that the FSAVE is executed. If the FPCP is in the reset state when the FSAVE instruction is received, only one word of state is transferred to memory, which may be examined by the operating system to determine that the coprocessor programmer's model is empty. If the coprocessor is idle when the save instruction is received, only a few words of internal state are transferred to memory. If the FPCP is in the middle of executing an instruction, it may be necessary to save the entire internal state of the machine. Instructions that can complete execution in less time than it would take to save the larger state in mid-instruction are allowed to complete execution and then save the idle state. Thus, the size of the saved internal state is kept to a minimum. The ability to utilize several internal state sizes greatly reduces the average context switching time.

The FRESTORE instruction permits reloading of an internal state that was saved earlier, and continues any operation that was previously suspended. Restoring of the reset internal state functions just like a hardware reset to the FPCP in that defaults are re-established.

5-5. Operand Data Formats

The FPCP supports the following data formats:

Byte Integer
Word Integer
Long Word Integer
Single Precision Real
Double Precision Real
Extended Precision Real
Packed Decimal String Real

5-6. Integer Data Formats

The three integer data formats (byte, word, and long word) are the standard data formats supported in the 68000 Family architecture. Whenever an integer is used in a floating-point operation, the integer is automatically converted by the FPCP to an extended precision floating-point number before being used. The ability to effectively use integers in floating-point operations saves user memory since an integer representation of a number, if representable, is usually smaller than the equivalent floating-point representation.

5-7. Floating-Point Data Formats

The floating-point data formats, single precision (32-bits) and double precision (64-bits) are as defined by the IEEE standard. These are the main floating-point formats and should be used for most calculations involving real numbers. Figure 5-7 lists the exponent and mantissa size for single, double, and extended precision. The exponent is biased, and the mantissa is in sign and magnitude form. Since single and double precision require normalized numbers, the most significant bit of the mantissa is implied as a one and is not included, thus giving one extra bit of precision.

Figure 5-7: Exponent And Mantissa Sizes

Data Format	Exponent Bits	Mantissa Bits
Single	8	23(+1)
Double	11	52(+1)
Extended	15	64

The extended precision data format is also in conformance with the IEEE standard, but the standard does not specify this format to the bit level as it does for single and double precision. The memory format on the FPCP consists of 96 bits (three long words). Only 80 bits are actually used, the other 16 bits are for future expandability and for long-word alignment of floating-point data structures. Extended format has a 15-bit exponent, a 64-bit mantissa, and a 1-bit mantissa sign.

5-8. Packed Decimal String Real Data Format

The packed decimal data format allows packed BCD strings to be input to and output from the FPCP. The strings consist of a 3-digit base 10 exponent and a 17-digit base 10 mantissa. Both the exponent and mantissa have a separate sign bit. All digits are packed BCD, such that an entire string fits in 96 bits (three long words). As is the case with all data formats, when packed BCD strings are input to the FPCP, the strings are automatically converted to extended precision real values. This allows packed BCD numbers to be used as inputs to any operation. BCD numbers can be output from the FPCP in a format readily used for printing by a program generated by a high-level language compiler.

5-9. Data Format Summary

All data formats described above are supported orthogonally by all arithmetic and transcendental operations, and by all appropriate 68020 addressing modes. For example, all of the following are legal instructions:

```
FADD.B    #0,FP0
FADD.W    D2,FP3
FADD.L    BIGINT,FP7
FADD.S    #3.14159,FP5
FADD.D    (SP)+,FP6
FADD.X    [(TEMP_PTR,A7)],FP3
FADD.P    #1.23E25,FP0
```

Most on-chip calculations are performed to extended precision format and the eight floating-point data registers always contain extended precision values. All data used in an operation is converted to extended precision by the FPCP before the specific operation is performed, and all results are in extended precision. This ensures maximum accuracy without sacrificing performance.

5-10. Floating-Point Coprocessor Instruction Set

The FPCP instruction set is organized into six major classes:

1. Moves between the FPCP and memory or the 68020 (in and out),
2. Move multiple registers (in and out),
3. Monadic operations,
4. Dyadic operations,
5. Branch, set, or trap conditionally, and
6. Miscellaneous.

5-10-1. Moves

On all moves from memory (or from a 68020 data register) to the FPCP, data is converted from the source data format to the internal extended precision format.

On all moves from the FPCP to memory (or to a 68020 data register), data is converted from the internal extended precision format to the destination data format.

Note that data movement instructions perform arithmetic operations, since the result is always rounded to the precision selected in the FPCR mode control byte. The result is rounded using the selected rounding mode and is checked for overflow and underflow.

5-10-2. Move Multiples

The floating-point move multiple instructions on the FPCP are much like the integer counterparts on the 68020 processor. Any set of the floating-point registers FP0 through FP7 can be moved to or from memory with one instruction. These registers are always moved as 96-bit extended data with no conversion (hence no possibility of conversion errors).

Move multiples are useful during context switches and interrupts to save or restore the state of a program. These moves are useful at the start and end of a procedure to save and restore the calling routine's register set. In order to reduce procedure call overhead, the list of registers to be saved or restored can be contained in a data register. This allows run-time optimization by allowing a called routine to save as few registers as possible. Note that no rounding or overflow/underflow checking is performed by these operations.

5-10-3. Monadic Operations

Monadic operations have one operand. This operand may be in a floating-point data register, memory, or in a 68020 data register. The result is always stored in a floating-point data register.

The monadic operations available with the FPCP are as follows:

FABS	Absolute Value	FLOG2	Log Base 2
FACOS	Arc Cosine	FLOGN	Log Base e
FASIN	Arc Sine	FLOGNP1	Log Base e of (x+1)
FATAN	Arc Tangent	FNEG	Negate
FATANH	Hyperbolic Arc Tangent	FSIN	Sine
FCOS	Cosine	FSINCOS	Simultaneous Sine & Cosine
FCOSH	Hyperbolic Cosine	FSINH	Hyperbolic Sine
FETOX	e to the x Power	FSQRT	Square Root
FETOXM1	e to the x Power -1	FTAN	Tangent
FGETEXP	Get Exponent	FTANH	Hyperbolic Tangent
FGETMAN	Get Mantissa	FTENTOX	10 to the x Power
FINT	Integer Part	FTST	Test
FINTRZ	Integer Part (Truncated)	FTWOTOX	2 to the x Power
FLOG10	Log Base 10		

5-10-4. Dyadic Operations

Dyadic operations have two operands. The first operand comes from a floating-point data register, memory, or a 68020 data register. The second operand comes from a floating-point data register. The destination is the same floating-point data register used for the second operand.

The FPCP dyadic operations available are as follows:

FADD	Add	FREM	IEEE Remainder
FCMP	Compare	FSCALE	Scale Exponent
FDIV	Divide	FSGLDIV	Single Precision Divide
FMOD	Modulo Remainder	FSGLMUL	Single Precision Multiply
FMUL	Multiply	FSUB	Subtract

Assuming that operands are single precision, the FSGLMUL and FSGLDIV instructions round results as such while maintaining the range of extended precision. In special applications where multiply and divide performance are more important than loss of precision, the FSGLMUL and FSGLDIV instructions can be used.

5-10-5. Branch, Set, And Trap-On Condition

The floating-point branch, set, and trap-on condition instructions implemented by the FPCP are similar to the equivalent integer instructions of the 68000 Family processors, except that more conditions exist due to the special values in IEEE floating-point arithmetic. When a conditional instruction is executed, the FPCP performs the necessary condition checking and tells the 68020 whether the condition is true or false; the 68020 then takes the appropriate action. Since the FPCP and 68020 are closely coupled, the floating-point branch operations are quickly executed.

The FPCP conditional operations are:

FBcc	Branch
FDBcc	Decrement and Branch
FScc	Set According to Condition
FTRAPcc	Trap-on Condition (with an Optional Parameter)

where:
cc is one of 32 floating-point conditional test specifiers.

5-10-6. Miscellaneous Instructions

Miscellaneous instructions include moves to and from the status, control, and instruction address registers. Also included are the virtual memory/machine FSAVE and FRESTORE instructions that save and restore the internal state of the FPCP.

FMOVE	Move to or from Control Register(s)
FSAVE	Virtual Machine State Save
FRESTORE	Virtual Machine State Restore

5-11. Floating-Point Coprocessor Addressing Modes

The FPCP does not perform address calculations. Thus, if the FPCP instructs the 68020 to transfer an operand via the coprocessor interface, the 68020 will perform the addressing mode calculations requested in the instruction.

This interface is quite flexible and allows any addressing mode to be used with floating-point instructions. These addressing modes include immediate, postincrement, predecrement, data or address register direct, and the indexed/indirect. Some addressing modes are restricted for some instructions in keeping with the 68000 Family architectural definitions (e.g., PC relative addressing is not allowed for a destination operand).

The orthogonal instruction set of the FPCP, along with the flexible branches and addressing modes, allows a programmer writing assembly language code, or a compiler writer generating object or source code for the 68020/FPCP device pair, to think of the FPCP as though the FPCP is part of the 68020. There are no special restrictions imposed by the coprocessor interface, and floating-point arithmetic is coded exactly like integer arithmetic.

5-12. 68882 Programming Considerations

To exploit the enhanced performance of the 68882 requires the programmer to be aware of the manner in which the coprocessor overlaps execution of instructions. Upgrading a system to use the 68882 requires minor system software changes but no user software changes. To optimize applications code for the 68882 may require reordering of floating-point instructions.

6. PROGRAMMING INFORMATION

The first half of Section 6 gives detailed information on the addresses and assignments of memory on the MPU-28 beginning with a summary table followed by a table for each I/O device.

The second half of Section 6 contains sample programs for the MPU-28.

6-1. Memory Addresses

The table below summarizes the addresses and assignments of memory on the MPU-28.

Figure 6-1: MPU-28 Memory Addresses

Address	Assignments
$00000000-$007FFFFF	On-Board Memory
$00800000-$00FFFFFF	Not Used
$01000000-$01FFFFFF	Multibus Memory Access (bytes are swapped)
$02000000-$03EFFFFF	Not Used
$03F00000-$03F3FFFF	28-Pin EPROM Sockets
$03F40000-$03F7FFFF	Not Used
$03F80000-$03F801FF	Translation Map 0, MPU-28 Accessed From Multibus
$03F80200-$03F803FF	Translation Map 1, MPU-28 Accessed From Multibus
$03F80400-$03F805FF	Translation Map 2, MPU-28 Accessed From Multibus
$03F80600-$03F807FF	Translation Map 3, MPU-28 Accessed From Multibus
$03F80800-$03F80FFF	On-Board I/O (see detailed figures in later sections)
$03F80801-$03F80807	8530 SCC Serial Ports
$03F80900-$03F80903	Control/Status Register
$03F80A00-**	iSBX Connector J20 Select 0
$03F80B00-**	iSBX Connector J20 Select 1
$03F80C00-$03F80DFF	SCSI Controller
$03F80E01-$03F80E7F	68230 Parallel Interface/Timer
$03F80F01	Watchdog Timer
$03F81000-$03F8FFFF	Not Used
$03F90000-$03F9FFFF	Multibus I/O (bytes are swapped)
$03FA0000-$04FFFFFF	Not Used
$05000000-$05FFFFFF	Multibus Memory Access (bytes are not swapped)
$06000000-$07F8FFFF	Not Used
$07F90000-$07F9FFFF	Multibus I/O (bytes are not swapped)
$07FA0000-$07FFFFFF	Not Used
$08000000-$08000001	MMU Control/Status Register
$18000000-$19FF0000	MMU Address Maps (32)

** Depends on Multimodule selected

6-2. Serial Port Addresses

Figure 6-2 shows the Read/Write registers associated with the I/O addresses of the 8530 Serial Communications Controller.

Figure 6-2
8530 Serial Communications Controller

Address	Read Register	Write Register
$03F80801	Ch B Status	Ch B Command
$03F80803	Ch B Data	Ch B Data
$03F80805	Ch A Status	Ch A Command
$03F80807	Ch A Data	Ch A Data

6-3. Parallel Interface/Timer Addresses

Figure 6-3 defines the registers of the 68230 Parallel Interface/Timer and gives their corresponding addresses.

Figure 6-3
68230 Parallel Interface/Timer

Address	Register
$03F80E01	Port General Control Register
$03F80E03	Port Service Request Register
$03F80E05	Port A Data Direction Register
$03F80E07	Port B Data Direction Register
$03F80E09	Port C Data Direction Register
$03F80E0B	Port Interrupt Vector Register
$03F80E0D	Port A Control Register
$03F80E0F	Port B Control Register
$03F80E11	Port A Data Register
$03F80E13	Port B Data Register
$03F80E15	Port A Alternate Register
$03F80E17	Port B Alternate Register
$03F80E19	Port C Data Register
$03F80E1B	Port Status Register
$03F80E1D	not used
$03F80E1F	not used
$03F80E21	Timer Control Register
$03F80E23	Timer Interrupt Vector Register
$03F80E25	not used
$03F80E27	Counter Pre-load Register (High)
$03F80E29	Counter Pre-load Register (Mid)
$03F80E2B	Counter Pre-load Register (Low)
$03F80E2D	not used
$03F80E2F	Count Register (High)
$03F80E31	Count Register (Mid)
$03F80E33	Count Register (Low)
$03F80E35	Timer Status Register
$03F80E37	not used
$03F80E39	not used
$03F8033B	not used
$03F80E3D	not used
$03F80E3F	not used

6-4. SCSI Addresses

Following is a list of the SCSI read and write registers and their respective addresses. Additional information on the use of these registers, with the exception of the DMA/data acknowledge register, can be found in the NCR 53C90 Enhanced SCSI Processor Data Sheet which is included at the end of this manual.

The DMA/data acknowledge register is discussed in Section 3-10-1 of this manual.

Figure 6-4: SCSI Register Address Assignments

Offset Address	Read Register	Write Register
$03F80C01	Transfer Counter Lo	Transfer Count Lo
$03F80C03	Transfer Counter Hi	Transfer Count Hi
$03F80C05	FIFO	FIFO
$03F80C07	Command	Command
$03F80C09	Status	Select/Reselect Bus ID
$03F80C0B	Interrupt Status	Select/Reselect Timeout
$03F80C0D	Sequence Step	Sync Period
$03F80C0F	FIFO Flags	Sync Offset
$03F80C11	Configuration	Configuration
$03F80C13	Reserved	Clock Conversion Factor
$03F80C15	Reserved	Test
$03F80D01	DMA/Data Acknowledge	DMA/Data Acknowledge

6-5. MPU-28 Sample Programs

This section contains listings of programming examples which may be used to supplement the information given in the manuals and data sheets on the I/O devices (see Section 9). All of the examples have been tested in actual running programs.

Intended to show the requirements for programming the I/O devices, these examples sacrifice optimal performance at times in order to be clear. If you choose to optimize these routines or otherwise alter them, be particularly careful to preserve the special timing requirements (e.g., inclusion of NOPs or "nonfunctional" loops).

```
*
*       M P U - 2 8   S A M P L E   P R O G R A M S
*

**      SERIAL PORT EQUATES

03F80805 CHANA   EQU     $03F80805       Address of Z8530 SCC channel A
03F80801 CHANB   EQU     $03F80801       Address of Z8530 SCC channel B
00000000 STATUS  EQU     0               Offset to SCC status port
00000000 CMD     EQU     0               Offset to SCC command port
00000002 DATA    EQU     2               Offset to SCC data port
00000000 INPUT   EQU     0               Rx status bit
00000002 OUTPUT  EQU     2               Tx status bit

**      CONTROL/STATUS REGISTER EQUATES

03F80900 SCR0    EQU     $03F80900       Control/status register (interrupt
                                         status)
00000080 SYSIR   EQU     $80             SYSIRQ status
00000040 LPERR   EQU     $40             Parity error status
00000020 LINSEL  EQU     $20             Mailbox interrupt status
00000010 ENBTF   EQU     $10             Watchdog timer status
00000008 DREQ    EQU     $08             DMA request status
00000004 SCSIRQ  EQU     $04             Interrupt status from SCSI
00000002 MINTR1  EQU     $02             Interrupt status from iSBX0
00000001 MINTR0  EQU     $01             Interrupt status from iSBX0

03F80901 SCR1    EQU     SCR0+1          Control/status register byte 1
00000080 READY   EQU     $80             Ready status to mailbox
00000040 MS1     EQU     $40             Map select
00000020 MS0     EQU     $20             Map select
00000010 BS      EQU     $10             Byte swap control
00000008 RPERR   EQU     $08             Parity error control
00000004 PARW    EQU     $04             Odd/even parity select
00000002 SYSIRQ  EQU     $02             Multibus interrupt output
00000001 RINSEL  EQU     $01             Mailbox interrupt control

03F80902 SCLED   EQU     SCR1+1          LED control byte 2 of control/status
                                         register
03F80902 SCINT   EQU     SCR1+1          Multibus interrupt status
```

```
          **      PARALLEL PORT/TIMER EQUATES

03F80E01 PIT      EQU     $03F80E01       Address of 68230 PI/T
00000000 PGCR     EQU     $00             Port General Control register
00000002 PSRR     EQU     $02             Port Service Request Register
00000004 PADDR    EQU     $04             Port A Data Direction Register
00000006 PBDDR    EQU     $06             Port B Data Direction Register
0000000C PACR     EQU     $0C             Port A Control Register
0000000E PBCR     EQU     $0E             Port B Control Register
00000010 PORTA    EQU     $10             Port A Data Register
00000012 PORTB    EQU     $12             Port B Data Register
00000018 PORTC    EQU     $18             Port C Data Register
0000001A PSR      EQU     $1A             Port Status register

00000000 PINPUT   EQU     0               Input ready status in PSR
00000002 POUTPUT  EQU     2               Output empty status in PSR

03F80E21 TIMER    EQU     PIT+$20         Address of 68230 timer
00000000 TIMERCNTRL EQU   0               Timer control word
00000002 TIMERVEC EQU     2               Interrupt vector number
00000004 TIMERPRE EQU     4               Counter preload
00000014 TIMERISTAT EQU   $14             Timer interrupt status

          **      MEMORY MAPPING PARAMETERS

03F80001 MAP      EQU     $03F80001       Address of shared memory
                                          translation map
00000800 MAPLEN   EQU     $0800           Size of map

          **      INTERRUPT VECTORS

000000C0 VECSIO   EQU     $C0             Serial I/O interrupt vectors (8
                                          vectors, 8 bytes apart)
000000C4 VECTMR   EQU     $C4             PI/T exception, timer interrupt

          **      BUFFER PARAMETERS

00004000 BUFFR1   EQU     $4000           Source buffer
00005000 BUFFR2   EQU     $5000           Destination buffer
00000200 BUFLEN   EQU     $200            Buffer length
```

```
                        **      Offsets within Channel Parameter Table (CHPTA and CHPTB)

000000                          ORG     0
000000                  INTACT  DS.B    1               Output interrupts active if
                                                        non-zero
000001                  IOFLAGS DS.B    1               Status flags
000002                  MRR0    DS.B    1               Saved copy of RR0
000003                  MRR1    DS.B    1               Saved copy of RR1
000004                  CHNUM   DS.W    1               Channel number (0-3)
000006                  CHAN    DS.L    1               Address of serial port

00000A                  RBEGIN  DS.L    1               Address of start of receive buffer
00000E                  REND    DS.L    1               Address of end+1 of receive buffer
000012                  RFILL   DS.L    1               Address of next character to fill
000016                  REMPTY  DS.L    1               Address of next character to empty
                                                        from buffer
                        *
00001A                  TBEGIN  DS.L    1               Address of start of transmit
                                                        buffer
00001E                  TEND    DS.L    1               Address of end+1 of transmit
                                                        buffer
000022                  TFILL   DS.L    1               Address of next character to fill
000026                  TEMPTY  DS.L    1               Address of next character to empty
                                                        from buffer


 )1000                          ORG     $1000
 01000                  SIOADR  DS.L    1               Serial port address

001004 00000000         TICKS   DC.L    0               Real-time clock
001008 00000000         COUNT   DC.L    0               Countdown clock
00100C 0000             ACKFLAG DC.W    0               Multibus interrupt acknowledge
                                                        flag
00100E 00FF             MASK    DC.W    $00FF           Multibus interrupt mask (1 bit per
                                                        interrupt line)
001010 000F             IMASK   DC.W    $0003           iSBX interrupt mask (1 bit per
                                                        interrupt line)

001012 00000000         CHPTA   DC.B    0,0,0,0         Channel A parameter table
001016 0000                     DC.W    0
001018 03F80805                 DC.L    CHANA
00101C                          DS.L    8

00103C 00000000         CHPTB   DC.B    0,0,0,0         Channel B parameter table
001040 0001                     DC.W    1
001042 03F80801                 DC.L    CHANB
001046                          DS.L    8
```

```
                        **      SDLC pointers and counts

001066                  DONE    DS.W    1               Interrupt status flag
001068                  EMPTYPTR DS.L   1               Used to read data from buffer
00106C                  FILLPTR DS.L    1               Used to write data to buffer
001070                  EMPTYCNT DS.W   1               Number of bytes left to read
001072                  FILLCNT DS.W    1               Amount of space left in write
                                                        buffer
001074                  STATR1  DS.B    1               Error status bits
001075                          DS.B    1



002000                          ORG     $2000

                        **      INITA - INITIALIZES CHANNEL A OF SCC

002000 41F90000208C     INITA   LEA     INIA,A0         Pointer to desired setup options
002006 203C00002580             MOVE.L  #9600,D0        Baud rate for channel A
00200C 45F903F80805             LEA     CHANA+CMD,A2    Address of channel A CMD register
002012 6000002E   =2042         BRA     INCH            Jump to finish channel setup



                        **      INITB - INITIALIZES CHANNEL B OF SCC

002016 41F90000208C     INITB   LEA     INIA,A0         Pointer to desired setup options
00201C 203C00002580             MOVE.L  #9600,D0        Baud rate for channel B
002022 45F903F80801             LEA     CHANB+CMD,A2    Address of channel B CMD register
002028 60000018   =2042         BRA     INCH            Jump to finish channel setup
```

```
                        **      SERVECS - SETUP SERIAL I/O INTERRUPT VECTORS
                        *
                        *       The SCC may be programmed to provide a separate interrupt
                        *       vector for each of the eight interrupt conditions.  This
                        *       routine sets all eight interrupt vectors to point to the
                        *       same interrupt service routine.  The interrupt routine
                        *       uses the vector offset on the stack to determine which
                        *       condition caused the interrupt.

00202C 43F800C0         SERVECS LEA     VECSIO,A1       Address of serial interrupt
                                                        vectors
002030 41F900002110             LEA     SCCINT,A0       Address of interrupt routine
002036 7007                     MOVE.L  #7,D0
002038 2288             SERVEC2 MOVE.L  A0,(A1)         Set up 8 vectors
00203A 5089                     ADD.L   #8,A1           Vectors are 8 bytes apart
00203C 51C8FFFA   =2038         DBRA    D0,SERVEC2
002040 4E75                     RTS



                        **      INCH - INITIALIZE CHANNEL: COMMON ROUTINE FOR CH. A AND B
                        *
                        *       INCH initializes a channel of the SCC using a table of
                        *       SCC register numbers and values.  By selecting the
                        *       desired table, you can set up a SCC channel for
                        *       non-interrupt I/O, for interrupts on input, output,
                        *       or both.
                        *
                        *       ENTRY (D0.W) = Desired baud rate (e.g., 300, 1200, 9600)
                        *             (A0)   = Address of table of desired setup options
                        *             (A2)   = Address of channel's command register
                        *

002042 1212             INCH    MOVE.B  (A2),D1         Ensure register pointing at WR0
002044 220A                     MOVE.L  A2,D1
002046 5641                     ADD.W   #3,D1
002048 E949                     LSL.W   #4,D1           Reset code for specified channel
00204A 14BC0009                 MOVE.B  #9,(A2)         Set register pointer for WR9
00204E 1481                     MOVE.B  D1,(A2)         Reset channel

002050 4241                     CLR.W   D1              Set up for DBRA instruction
002052 1218                     MOVE.B  (A0)+,D1        First table byte = # of entries -
                                                        1
002054 1498             INCH1   MOVE.B  (A0)+,(A2)      Setup byte to SCC
002056 51C9FFFC   =2054         DBRA    D1,INCH1

00205A 223C0001C200             MOVE.L  #115200,D1      3.6864MHz/(2 * 16)
002060 82C0                     DIVU    D0,D1           Calculate BR generator interval
002062 5541                     SUB.W   #2,D1           Baud Rate time constant
```

```
002064 14BC000E                MOVE.B  #14,(A2)        Set for BR generator register WR14
002068 14BC0080                MOVE.B  #$80,(A2)       Disable Baud Rate generator
00206C 14BC000C                MOVE.B  #12,(A2)        Set register pointer to WR12
002070 1481                    MOVE.B  D1,(A2)         Set Baud Rate (low byte)
002072 14BC000D                MOVE.B  #13,(A2)        Set register pointer to WR13
002076 E049                    LSR.W   #8,D1
002078 1481                    MOVE.B  D1,(A2)         Set Baud Rate (high byte)
00207A 14BC000E                MOVE.B  #14,(A2)        Set for BR generator register WR14
00207E 14BC0003                MOVE.B  #$03,(A2)       Enable Baud Rate generator

002082 323C2000                MOVE.W  #$2000,D1       Allow initialization to settle
002086 5341            INCH2   SUB.W   #1,D1
002088 66FC       =2086        BNE     INCH2           Loop
00208A 4E75                    RTS                     Exit from channel initialization



             0000208C INIA     EQU     *               NON-INTERRUPT INITIALIZATION
                      *                                WITHOUT RTS & CTS AUTO ENABLES
00208C 09                      DC.B    INIAE-INIA-2    # initialization bytes - 1
00208D 0100                    DC.B    1,0             WR1, Clear Tx and Rx enables
00208F 0444                    DC.B    4,$44           WR4, *16 clock/1 stop bit per char
002091 03C1                    DC.B    3,$C1           WR3, 8 bits/Rx enable
002093 05E8                    DC.B    5,$E8           WR5, DTR/8 bits per char/Tx enable
002095 0B56                    DC.B    11,$56          WR11, Rx,Tx clk/TRxC
                                                       out/TRxC=BRgen
             00002097 INIAE    EQU     *

             00002097 INIB     EQU     *               VECTORED INTERRUPT INITIALIZATION
                      *                                WITH RTS & CTS AUTO ENABLES
002097 0F                      DC.B    INIBE-INIB-2    # initialization bytes -1
002098 0100                    DC.B    1,0             WR1, Clear Tx and Rx enables
00209A 0444                    DC.B    4,$44           WR4, *16 clock/1 stop bit per char
00209C 03E1                    DC.B    3,$E1           WR3, 8 bits/AUTO ENABLES/Rx enable
00209E 05EA                    DC.B    5,$EA           WR5, DTR/8 bits/Tx enable/RTS
0020A0 0230                    DC.B    2,VECSIO/4      WR2, Interrupt vector base = 0
0020A2 0909                    DC.B    9,$09           WR9, MIE/Vector includes
                                                       status/Status Low
0020A4 0B56                    DC.B    11,$56          WR11, Rx,Tx clk/TRxC
                                                       out/TRxC=BRgen
0020A6 0112                    DC.B    1,$12           WR1, Enable Rx and Tx interrupts
             000020A8 INIBE    EQU     *
```

```
                        **      RCA - READ CHARACTER FROM CHANNEL A
                        *
                        *       This routine assumes the SCC character I/O
                        *       is not interrupt driven.  The status port
                        *       of the SCC is polled until a character is
                        *       available.
                        *
                        *       EXIT  (D0) = CHARACTER READ FROM SCC
                        *

0020A8 0839000003F80805 RCA     BTST    #INPUT,CHANA+STATUS Wait on character received
0020B0 67F6       =20A8         BEQ     RCA
0020B2 103903F80807             MOVE.B  CHANA+DATA,D0     Read character
0020B8 C07C007F                 AND.W   #$7F,D0           Ignore input device's parity bit
                        *
                        *       Drop through to echo the character
                        *



                        **      WCA - WRITE CHARACTER TO CHANNEL A
                        *
                        *       This routine assumes the SCC character I/O
                        *       is not interrupt-driven.  The status port
                        *       of the SCC is polled until the transmit
                        *       port FIFO will accept the next character.
                        *
                        *       ENTRY (D0) = CHARACTER TO WRITE
                        *

0020BC 0839000203F80805 WCA     BTST    #OUTPUT,CHANA+STATUS Wait on transmitter empty
0020C4 67F6       =20BC         BEQ     WCA
0020C6 13C003F80807             MOVE.B  D0,CHANA+DATA     Output character
0020CC 4E75                     RTS
```

```
                        **      OUTCHAR - OUTPUT OR BUFFER CHARACTER
                        *
                        *       This is the character output routine that sets up
                        *       the output buffer for the interrupt routine (SCCINT).
                        *       The SCC only interrupts after a character has been
                        *       output, so the first character has to be handled
                        *       from outside the interrupt routine.
                        *
                        *       The first character is sent and INTACT is set.
                        *       Thereafter, characters are buffered (to be output
                        *       by SCCINT).  When SCCINT empties the buffer it clears
                        *       INTACT.
                        *
                        *       ENTRY (D0) = BYTE TO OUTPUT
                        *             (A1) = ADDRESS OF CHANNEL PARAMETER TABLE
                        *

0020CE 2F08             OUTCHAR MOVE.L  A0,-(A7)
0020D0 4A11                     TST.B   INTACT(A1)
0020D2 672C     =2100           BEQ.S   OUTCH6          Jump if SCC interrupts not active

0020D4 20690022                 MOVE.L  TFILL(A1),A0
0020D8 10C0                     MOVE.B  D0,(A0)+        Put character in buffer
0020DA B1E9001E                 CMP.L   TEND(A1),A0
0020DE 6604     =20E4           BNE.S   OUTCH2
0020E0 2069001A                 MOVE.L  TBEGIN(A1),A0
0020E4 B1E90026         OUTCH2  CMP.L   TEMPTY(A1),A0   If buffer full, wait until not
                                                        full
0020E8 67FA     =20E4           BEQ     OUTCH2

0020EA 40E7                     MOVE    SR,-(A7)        Save interrupt level
0020EC 007C0700                 OR      #$700,SR        Turn interrupts off
0020F0 4A11                     TST.B   INTACT(A1)
0020F2 670A     =20FE           BEQ.S   OUTCH4          Jump if SCC interrupts went
                                                        inactive
0020F4 23480022                 MOVE.L  A0,TFILL(A1)    Update buffer pointer
0020F8 46DF                     MOVE    (A7)+,SR        Restore interrupts
0020FA 205F                     MOVE.L  (A7)+,A0
0020FC 4E75                     RTS

0020FE 46DF             OUTCH4  MOVE    (A7)+,SR        Restore interrupts
002100 12BC0001         OUTCH6  MOVE.B  #1,INTACT(A1)   Flag interrupts active
002104 20690006                 MOVE.L  CHAN(A1),A0
002108 11400002                 MOVE.B  D0,DATA(A0)     Output the byte
00210C 205F                     MOVE.L  (A7)+,A0
00210E 4E75                     RTS
```

```
                        **      SCCINT - SCC INTERRUPT PROCESSING
                        *
                        *       This interrupt routine uses the vector offset value
                        *       on the stack to index through a jump table to
                        *       process the appropriate type of interrupt.
                        *       The vector is supplied by the SCC CHANNEL B's
                        *       RR2 read register which is modified by the SCC
                        *       to indicate the cause of the interrupt.
                        *

002110 48E780E0         SCCINT  MOVEM.L D0/A0/A1/A2,-(A7) Save registers
002114 41F903F80801             LEA     CHANB,A0          Address of serial port B
00211A 7038                     MOVE.L  #$38,D0
00211C C06F0016                 AND.W   4*4+6(A7),D0      Get vector number from stack
002120 43F8103C                 LEA     CHPTB,A1          Address of channel B parameter
                                                          table
002124 45FA0010   =2136         LEA     CASETB(PC),A2     Base address of jump table
002128 4EB20000                 JSR     (A2,D0.W)         Get address of interrupt routine

00212C 10BC0038                 MOVE.B  #$38,CMD(A0)      WR0 command to reset highest IUS
002130 4CDF0701                 MOVEM.L (A7)+,D0/A0/A1/A2 Restore registers
002134 4E73                     RTE



                        **      CASETB - INTERRUPT TYPE TABLE

002136 6000006A   =21A2 CASETB  BRA.L   OUTBINT           CH B transmit buffer empty
00213A 00000000                 DC.L    0
00213E 6000009A   =21DA         BRA.L   BXSINT           CH B external/status change
002142 00000000                 DC.L    0
002146 60000030   =2178         BRA.L   INBINT           CH B receive character available
00214A 00000000                 DC.L    0
00214E 600000B0   =2200         BRA.L   BSRINT           CH B special receive condition
002152 00000000                 DC.L    0
002156 60000044   =219C         BRA.L   OUTAINT          CH A transmit buffer empty
00215A 00000000                 DC.L    0
00215E 60000074   =21D4         BRA.L   AXSINT           CH A external/status change
002162 00000000                 DC.L    0
002166 6000000A   =2172         BRA.L   INAINT           CH A receive character available
00216A 00000000                 DC.L    0
00216E 6000008A   =21FA         BRA.L   ASRINT           CH A special receive condition
```

```
                        **      INAINT - PROCESS INPUT INTERRUPT ON CHANNEL A
                        **      INBINT - PROCESS INPUT INTERRUPT ON CHANNEL B
                        *

002172 43F81012         INAINT  LEA     CHPTA,A1        Address of channel A parameter
                                                        table
002176 5848                     ADD.W   #4,A0           Point to channel A

002178 24690012         INBINT  MOVE.L  RFILL(A1),A2
00217C 14E80002         INBIN2  MOVE.B  DATA(A0),(A2)+  Read character and save in buffer

002180 B5E9000E                 CMP.L   REND(A1),A2
002184 6604       =218A         BNE.S   INBIN4
002186 2469000A                 MOVE.L  RBEGIN(A1),A2   Wrap-around to start of buffer
00218A B5E90016         INBIN4  CMP.L   REMPTY(A1),A2
00218E 670A       =219A         BEQ.S   INBIN6          Buffer overflow
002190 234A0012                 MOVE.L  A2,RFILL(A1)
002194 08100000                 BTST    #INPUT,STATUS(A0)
002198 66E2       =217C         BNE.S   INBIN2          Next character is waiting
00219A 4E75             INBIN6  RTS                     Return to SCCINT
```

```
                        **      OUTAINT - PROCESS OUTPUT-EMPTY INTERRUPT ON CHANNEL A
                        **      OUTBINT - PROCESS OUTPUT-EMPTY INTERRUPT ON CHANNEL B
                        *
                        *       If no more characters are available for output on
                        *       an output interrupt, the output interrupts are
                        *       turned off.  They are automatically started again
                        *       when a character is output to the appropriate port
                        *       by the background program (see OUTCHAR example).  An
                        *       output-interrupts-active flag must be maintained to
                        *       avoid output conflicts.
                        *

00219C 43F81012         OUTAINT LEA     CHPTA,A1        Address of channel A parameter
                                                        table
0021A0 5848                     ADD.W   #4,A0           Point to channel A

0021A2 24690026         OUTBINT MOVE.L  TEMPTY(A1),A2
0021A6 B5E90022                 CMP.L   TFILL(A1),A2
0021AA 6720       =21CC         BEQ.S   OUTBI8          Circular buffer is empty

0021AC 115A0002         OUTBI2  MOVE.B  (A2)+,DATA(A0)  Write character to serial port
0021B0 B5E9001E                 CMP.L   TEND(A1),A2
0021B4 6604       =21BA         BNE.S   OUTBI4
0021B6 2469001A                 MOVE.L  TBEGIN(A1),A2   Wrap around to start of buffer
0021BA 08100002         OUTBI4  BTST    #OUTPUT,STATUS(A0)
 021BE 6706       =21C6         BEQ.S   OUTBI6          Channel cannot accept another
                        *                               character
0021C0 B5E90022                 CMP.L   TFILL(A1),A2
0021C4 66E6       =21AC         BNE.S   OUTBI2          Get next character from circular
                        *                               buffer



0021C6 234A0026         OUTBI6  MOVE.L  A2,TEMPTY(A1)
0021CA 4E75                     RTS                     Return to SCCINT

                        *                               If no more CHANNEL B output
0021CC 10BC0028         OUTBI8  MOVE.B  #$28,CMD(A0)    Turn off OUTPUT interrupts
0021D0 4211                     CLR.B   INTACT(A1)      Flag interrupts off
0021D2 4E75                     RTS                     Return to SCCINT
```

```
                        **      AXSINT - PROCESS EXTERNAL/STATUS CHANGE INTERRUPT ON CH A
                        **      BXSINT - PROCESS EXTERNAL/STATUS CHANGE INTERRUPT ON CH B
                        *

0021D4 43F81012         AXSINT  LEA     CHPTA,A1        Address of channel A parameter
                                                        table
0021D8 5848                     ADD.W   #4,A0           Point to channel A

0021DA 1010             BXSINT  MOVE.B  STATUS(A0),D0
0021DC B1290002                 EOR.B   D0,MRR0(A1)
0021E0 10BC000F                 MOVE.B  #15,CMD(A0)
0021E4 1010                     MOVE.B  STATUS(A0),D0   Read RR15 (external status
                                                        interrupt enables)
                        *
0021E6 C0290002                 AND.B   MRR0(A1),D0     Mask off unused bits
                        *        .
                        *        .                      D0 contains changed bits
                        *        .
0021EA 13500002                 MOVE.B  STATUS(A0),MRR0(A1) Update RR0 in memory
0021EE 002900020001             OR.B    #2,IOFLAGS(A1)  Flag interrupt as occurring
0021F4 10BC0010                 MOVE.B  #$10,CMD(A0)    Reset external/status interrupt
0021F8 4E75                     RTS                     Return to SCCINT



                        **      ASRINT - PROCESS SPECIAL RECEIVE COND. INTERRUPT ON CH A
                        **      BSRINT - PROCESS SPECIAL RECEIVE COND. INTERRUPT ON CH B
                        *

0021FA 43F81012         ASRINT  LEA     CHPTA,A1        Address of channel A parameter
                                                        table
0021FE 5848                     ADD.W   #4,A0           Point to channel A

002200 10BC0001         BSRINT  MOVE.B  #1,CMD(A0)
002204 1010                     MOVE.B  STATUS(A0),D0   Read RR1 (error status)
002206 13400003                 MOVE.B  D0,MRR1(A1)     Save error status
                        *        .
                        *        .                      Interrupt error processing
                        *        .
00220A 002900010001             OR.B    #1,IOFLAGS(A1)  Flag interrupt as occurring
002210 10BC0030                 MOVE.B  #$30,CMD(A0)    Reset error
002214 4E75                     RTS                     Return to SCCINT
```

```
                        **      SDLC - SEND AND RECEIVE SDLC FRAMES
                        *
                        *       This program transmits and receives data in SDLC
                        *       mode.  It is assumed that the transmit data and
                        *       clock signals of channel A have been looped back
                        *       to the receive data and clock signals of channel A.
                        *
                        *       This routine repeats indefinitely and only exits
                        *       if an error is detected.
                        *

002216 41F84000         SDLC    LEA     BUFFR1,A0
00221A 303C01FF                 MOVE.W  #BUFLEN-1,D0
00221E 4281                     CLR.L   D1
002220 10C1             SDLC2   MOVE.B  D1,(A0)+        Initialize source buffer
002222 5241                     ADD.W   #1,D1
002224 51C8FFFA   =2220         DBRA    D0,SDLC2

002228 61000074   =229E         BSR     SDLCVECS        Set up interrupt vectors
00222C 42B81008                 CLR.L   COUNT

002230 41F903F80805             LEA     CHANA,A0
002236 21C81000                 MOVE.L  A0,SIOADR
00223A 43FA00B4   =22F0         LEA     SCCTBL(PC),A1
00223E 6100009C   =22DC         BSR     INSER           Configure serial port

 02242 027CF8FF                 AND     #$F8FF,SR

002246 41F85000         SDLC4   LEA     BUFFR2,A0
00224A 303C01FF                 MOVE.W  #BUFLEN-1,D0
00224E 72FF                     MOVE.L  #-1,D1
002250 10C1             SDLC4A  MOVE.B  D1,(A0)+        Blank out destination buffer
002252 51C8FFFC   =2250         DBRA    D0,SDLC4A

002256 21FC000040001068         MOVE.L  #BUFFR1,EMPTYPTR Initialize pointers and counts
00225E 31FC02001070             MOVE.W  #BUFLEN,EMPTYCNT
002264 21FC00005000106C         MOVE.L  #BUFFR2,FILLPTR
00226C 31FC02041072             MOVE.W  #BUFLEN+4,FILLCNT

002272 52B81008                 ADD.L   #1,COUNT

002276 20781068                 MOVE.L  EMPTYPTR,A0
00227A 1018                     MOVE.B  (A0)+,D0        Get first character
00227C 21C81068                 MOVE.L  A0,EMPTYPTR

002280 20781000                 MOVE.L  SIOADR,A0
002284 11400002                 MOVE.B  D0,DATA(A0)     Output first character
002288 42781066                 CLR.W   DONE

00228C 0C7800031066     SDLC8   CMP.W   #3,DONE         Wait for transmit and receive to
                                                        finish
002292 66F8       =228C         BNE     SDLC8
```

```
002294 6100002E   =22C4          BSR     VERIFY          Verify results
002298 6702       =229C          BEQ.S   SDLC9
00229A 4E40                      TRAP    #0              Error detected
00229C 60A8       =2246 SDLC9    BRA     SDLC4



                        **       SDLCVECS - SETUP SERIAL I/O INTERRUPT VECTORS
                        *
                        *        The SCC may be programmed to provide a separate interrupt
                        *        vector for each of the eight interrupt conditions.  This
                        *        routine sets the channel A interrupt vectors to point to
                        *        the SDLC interrupt processing routines.
                        *

00229E 43F800E0          SDLCVECS LEA    VECSIO+4*8,A1   Address of serial interrupt
                                                         vectors
0022A2 41F9000022B4              LEA     SERTBL,A0       Address of interrupt routine
0022A8 7003                      MOVE.L  #3,D0
0022AA 2298              SDLCV2  MOVE.L  (A0)+,(A1)      Set up 8 vectors
0022AC 5089                      ADD.L   #8,A1           Vectors are 8 bytes apart
0022AE 51C8FFFA   =22AA          DBRA    D0,SDLCV2
0022B2 4E75                      RTS

                        **       SERTBL - SERIAL INTERRUPT ADDRESS TABLE

0022B4 0000233E          SERTBL  DC.L    SDLCOUT         CH A transmit buffer empty
0022B8 0000238A                  DC.L    SDLCXS          CH A external/status change
0022BC 0000231C                  DC.L    SDLCIN          CH A receive character available
0022C0 0000239C                  DC.L    SDLCEOF         CH A special receive condition



                        **       VERIFY - VERIFY DESTINATION BUFFER
                        *
                        *        EXIT (D0.W) = 0 IF BUFFERS MATCH
                        *

0022C4 41F84000          VERIFY  LEA     BUFFR1,A0
0022C8 43F85000                  LEA     BUFFR2,A1
0022CC 303C0200                  MOVE.W  #BUFLEN,D0
0022D0 B308              VER2    CMP.B   (A0)+,(A1)+
0022D2 66000006   =22DA          BNE     VER4
0022D6 5340                      SUB.W   #1,D0
0022D8 66F6       =22D0          BNE     VER2
0022DA 4E75              VER4    RTS
```

```
                        **      INSER - INITIALIZE SCC SERIAL PORT
                        *
                        *       ENTRY (A0) = ADDRESS OF SERIAL PORT
                        *             (A1) = ADDRESS OF INITIALIZATION TABLE
                        *

0022DC 1010             INSER   MOVE.B  (A0),D0         Set pointer to register 0
0022DE 3219                     MOVE.W  (A1)+,D1        Get byte count

0022E0 1099             INSER2  MOVE.B  (A1)+,(A0)      Send byte to SCC
0022E2 5341                     SUB.W   #1,D1
0022E4 66FA       =22E0         BNE     INSER2

0022E6 323C2000                 MOVE.W  #$2000,D1
0022EA 5341             INSER4  SUB.W   #1,D1           Pause after serial port
                                                        configuration
0022EC 66FC       =22EA         BNE     INSER4
0022EE 4E75                     RTS



                        **      SCCTBL - SCC INITIALIZATION TABLE

0022F0 002A             SCCTBL  DC.W    SCCT2-SCCT1
               000022F2 SCCT1   EQU     *
0022F2 0420                     DC.B    $04,$20         X1 clock, SDLC mode
0022F4 05E2                     DC.B    $05,$E2         DTR, Tx 8 bits, RTS
0022F6 03C0                     DC.B    $03,$C0         Rx 8 bits, hunt mode
0022F8 0100                     DC.B    $01,$00         Disable interrupts
0022FA 0230                     DC.B    $02,VECSIO/4    Interrupt vector
0022FC 0600                     DC.B    $06,$00         Frame address
0022FE 077E                     DC.B    $07,$7E         Sync character
002300 0B16                     DC.B    $0B,$16         Rx clock = RTXC* pin, Tx clock =
                                                        BR generator
002302 0E80                     DC.B    $0E,$80         Disable BR generator
002304 0C16                     DC.B    $0C,$16         Baud rate (lower) 76.8K BAUD
002306 0D00                     DC.B    $0D,$00         Baud rate (upper)
002308 0E83                     DC.B    $0E,$83         Enable BR generator
00230A 0113                     DC.B    $01,$13         Enable Rx interrupts on all,
                                                        enable Tx interrupts
00230C 0A80                     DC.B    $0A,$80         CRC preset
00230E 0F40                     DC.B    $0F,$40         EOM interrupt enable
002310 10                       DC.B    $10             Reset external status interrupt
002311 10                       DC.B    $10
002312 28                       DC.B    $28             Reset Tx interrupt
002313 30                       DC.B    $30             Reset error
002314 38                       DC.B    $38             Reset highest IUS
002315 03C1                     DC.B    $03,$C1         Rx enable
002317 05EB                     DC.B    $05,$EB         Enable Tx
002319 80                       DC.B    $80             Reset CRC generator
00231A 0909                     DC.B    $09,$09         Enable interrupts
               0000231C SCCT2   EQU     *
```

```
                        **      SDLCIN - PROCESS SDLC INPUT INTERRUPT
                        *

00231C 11B903F8080701E1 SDLCIN  MOVE.B  CHANA+DATA,([FILLPTR]) Get input and save in buffer
002326 52B8106C                 ADD.L   #1,FILLPTR       Update buffer pointer
00232A 53781072                 SUB.W   #1,FILLCNT       Update counter
00232E 6700000C   =233C         BEQ     INTERR
002332 13FC003803F80805         MOVE.B  #$38,CHANA+CMD   Reset highest IUS
00233A 4E73                     RTE

00233C 4E42             INTERR  TRAP    #2               Buffer overflow



                        **      SDLCOUT - PROCESS SDLC OUTPUT-EMPTY INTERRUPT
                        *
                        *       If no more characters are available for output on
                        *       an output interrupt, the output interrupts are
                        *       turned off.  They are automatically started again
                        *       when a character is output to channel A by the
                        *       background program.
                        *

00233E 53781070         SDLCOUT SUB.W   #1,EMPTYCNT       Update character counter
002342 6F00001A   =235E         BLE     SDLCO2           Buffer is empty
002346 13F001E1106803F8         MOVE.B  ([EMPTYPTR]),CHANA+DATA Output next character
002350 52B81068                 ADD.L   #1,EMPTYPTR      Advance buffer pointer
002354 13FC003803F80805         MOVE.B  #$38,CHANA+CMD   Reset highest IUS
00235C 4E73                     RTE

00235E 6B0A       =236A SDLCO2  BMI.S   SDLCO4           Branch if end of CRC
002360 13FC00C003F80805         MOVE.B  #$C0,CHANA+CMD   Clear Tx underrun latch
002368 600E       =2378         BRA.S   SDLCO6

00236A 13FC008003F80805 SDLCO4  MOVE.B  #$80,CHANA+CMD   Reset Tx CRC generator
002372 007800021066             OR.W    #$02,DONE        Flag transmit frame complete
002378 13FC002803F80805 SDLCO6  MOVE.B  #$28,CHANA+CMD   Reset Tx interrupt
002380 13FC003803F80805         MOVE.B  #$38,CHANA+CMD   Reset highest IUS
002388 4E73                     RTE
```

```
                        **      SDLCXS - PROCESS SDLC EXTERNAL/STATUS CHANGE INTERRUPT
                        *
                        *       This interrupt indicates that the frame being sent
                        *       is complete and the CRC is about to be sent.
                        *

00238A 13FC001003F80805 SDLCXS  MOVE.B  #$10,CHANA+CMD    Reset interrupt
002392 13FC003803F80805         MOVE.B  #$38,CHANA+CMD    Reset highest IUS
00239A 4E73                     RTE



                        **      SDLCEOF - PROCESS SDLC SPECIAL RECEIVE COND. INTERRUPT
                        *
                        *       When the frame being read is complete, this interrupt
                        *       is generated and the receive status is available in
                        *       read register 1.
                        *

00239C 13FC000103F80805 SDLCEOF MOVE.B  #1,CHANA+CMD
0023A4 11F903F808051074         MOVE.B  CHANA+STATUS,STATR1 Save error status (read
                                                              register 1)
0023AC 023800F01074             AND.B   #$F0,STATR1
0023B2 0C3800801074             CMP.B   #$80,STATR1
0023B8 6618       =23D2         BNE.S   SDLCRE              Frame error

0023BA 007800011066             OR.W    #1,DONE             Flag interrupt as occurring
0023C0 13FC003003F80805         MOVE.B  #$30,CHANA+CMD      Reset error
0023C8 13FC003803F80805         MOVE.B  #$38,CHANA+CMD      Reset highest IUS
0023D0 4E73                     RTE

0023D2 4E41             SDLCRE  TRAP    #1                  Receive error
```

```
                    **      TIMERSETUP - TIMER SETUP ROUTINE
                    *
                    *       This is the initialization routine for the timer
                    *       interrupt routine (TIMERINT).
                    *

            000023D4 TIMERSETUP EQU  *
0023D4 41F903F80E21          LEA     TIMER,A0
0023DA 10BC00A0              MOVE.B  #$A0,TIMERCNTRL(A0) Disable the timer
0023DE 117C00310002          MOVE.B  #VECTMR/4,TIMERVEC(A0) Use exception vector at $D0

0023E4 21FC0000240800C4      MOVE.L  #TIMERINT,VECTMR Initialize exception vector
0023EC 42B81004              CLR.L   TICKS            Clear timer associated variables
0023F0 42B81008              CLR.L   COUNT

0023F4 203C00007A11          MOVE.L  #1000000/32-1,D0 Clock interval for 1/10 second
0023FA 01C80004              MOVEP.L D0,TIMERPRE(A0)
0023FE 10BC00A1              MOVE.B  #$A1,TIMERCNTRL(A0) Enable the timer

002402 027CF8FF              AND     #$F8FF,SR        Enable interrupts
002406 4E75                  RTS




                    **      TIMERINT - TIMER INTERRUPT ROUTINE
                    *
                    *       The timer interrupts every 1/10th of a second.
                    *       The timer interrupt is a level 1 vectored interrupt
                    *       that uses the vector at 'VECTMR'.
                    *
                    *       This interrupt routine updates two timers, TICKS and
                    *       COUNT.  TICKS advances continuously every 1/10th of
                    *       a second.  COUNT counts down to zero and then stops
                    *       counting.
                    *

            00002408 TIMERINT EQU    *
002408 2F08                  MOVE.L  A0,-(A7)
00240A 41F903F80E21          LEA     TIMER,A0
002410 117C00010014          MOVE.B  #1,TIMERISTAT(A0) Clear the interrupt

002416 52B81004              ADD.L   #1,TICKS         Advance real-time clock
00241A 4AB81008              TST.L   COUNT
00241E 6704       =2424      BEQ.S   TIMRI2
002420 53B81008              SUB.L   #1,COUNT         Decrement countdown timer
002424 205F            TIMRI2 MOVE.L  (A7)+,A0
002426 4E73                  RTE
```

```
                        **      PITSETUP - Initialize PI/T parallel port
                        *
                        *       Two MPU-28 boards may communicate with each other via
                        *       the 68230 PI/T.  The PI/T performs all of the handshaking
                        *       and synchronization necessary for proper communication
                        *       between two boards.
                        *
                        *       The boards are assumed to be connected as follows:
                        *
                        *       board 1   board 2
                        *       -------   -------
                        *       Port A <= Port B
                        *       Port B => Port A
                        *       H1, H3 <= H4, H2
                        *       H4, H2 => H1, H3
                        *

002428 41F903F80E01     PITSETUP LEA    PIT,A0
00242E 43F900002444              LEA    PITTBL,A1
002434 3419                      MOVE.W (A1)+,D2          Number of parameters to load

002436 3019             PITS2    MOVE.W (A1)+,D0          PI/T register to load
002438 3219                      MOVE.W (A1)+,D1          PI/T register contents
00243A 11810000                  MOVE.B D1,(A0,D0.W)      Load register
00243E 5542                      SUB.W  #2,D2
002440 66F4       =2436          BNE.S  PITS2

002442 4E75                      RTS



                        **      PITTBL - PARALLEL PORT PARAMETER TABLE

              00002444 PITTBL   EQU    *
002444 0018                     DC.W   PITTE-PITTBL-2   Number of registers to initialize
002446 0000003F                 DC.W   $00,$3F          PGCR  - Mode 0, handshake lines
                                                        enabled
00244A 00020018                 DC.W   $02,$18          PSRR  - Vectored interrupts
00244E 00040000                 DC.W   $04,$00          PADDR - A port is input
002452 000600FF                 DC.W   $06,$FF          PBDDR - B port is output
002456 000C0030                 DC.W   $0C,$30          PACR  - Submode 00, interlocked
                                                        handshake
00245A 000E0070                 DC.W   $0E,$70          PBCR  - Submode 01, interlocked
                                                        handshake
              0000245E PITTE    EQU    *                End of table
```

```
                        **      PREAD - READ DATA FROM PARALLEL PORT

00245E 0839000003F80E1B PREAD   BTST    #PINPUT,PIT+PSR
002466 67F6       =245E         BEQ.S   PREAD           Wait for data from PI/T
002468 103903F80E11             MOVE.B  PIT+PORTA,D0    Read data
00246E 4E75                     RTS
                        **      PWRITE - WRITE DATA TO PARALLEL PORT

002470 0839000203F80E1B PWRITE  BTST    #POUTPUT,PIT+PSR
002478 67F6       =2470         BEQ.S   PWRITE          Wait for PI/T to be ready
00247A 13C003F80E13              MOVE.B  D0,PIT+PORTB    Write data
002480 4E75                     RTS




                        **      MAILBOX - PROCESS MAILBOX INTERRUPT
                        *
                        *       When another Multibus master writes to the MPU-28's
                        *       Multibus I/O address, an interrupt is generated.
                        *

002482 023900FE03F80901 MAILBOX AND.B   #$FF-RINSEL,SCR1 Remove interrupt
00248A 0039000103F80901         OR.B    #RINSEL,SCR1    Re-enable mailbox interrupts
                        *        .
                        *        .            Perform "mailbox" interrupt functions
                        *        .
002492 4E73                     RTE
```

```
                        **      MBINT - Process Multibus or iSBX interrupt
                        *
                        *       The 8 Multibus and 2 iSBX interrupts all generate
                        *       level 2 auto vectored interrupts.  This routine
                        *       determines the source of the interrupt and passes
                        *       control to the appropriate service routine.
                        *

002494 48E7E0E0         MBINT   MOVEM.L D0-D2/A0-A2,-(A7)
002498 103903F80902             MOVE.B  SCINT,D0          Get status of Multibus interrupt
                                                          lines
00249E C078100E                 AND.W   MASK,D0           Filter out unused interrupt lines

0024A2 41F9000024F6             LEA     INTPRI,A0
0024A8 10300000                 MOVE.B  0(A0,D0.W),D0     Select highest priority interrupt
                                                          to process
0024AC 4EB000DC                 JSR     INTTBL-INTPRI(A0,D0.W) Go to interrupt processing
                                                                routine
0024B0 4CDF0707                 MOVEM.L (A7)+,D0-D2/A0-A2
0024B4 4E73                     RTE



0024B6 103903F80900     ISBXINT MOVE.B  SCR0,D0           Get status of iSBX interrupt lines
0024BC C0781010                 AND.W   IMASK,D0          Filter out unused interrupt lines

 024C0 41F90000260A             LEA     SBXPRI,A0
0024C6 10300000                 MOVE.B  0(A0,D0.W),D0     Select highest priority interrupt
                                                          to process
0024CA 4EF000EC                 JMP     SBXTBL-SBXPRI(A0,D0.W) Go to interrupt processing
                                                                routine
```

```
                000024CE INT0     EQU     *               Service routine for INT0*
                000024CE INT1     EQU     *               Service routine for INT1*
                000024CE INT2     EQU     *               Service routine for INT2*
                000024CE INT3     EQU     *               Service routine for INT3*
                000024CE INT4     EQU     *               Service routine for INT4*
                000024CE INT5     EQU     *               Service routine for INT5*
                000024CE INT6     EQU     *               Service routine for INT6*
                000024CE INT7     EQU     *               Service routine for INT7*
0024CE 4E75                       RTS

                000024D0 MIN0     EQU     *               Service routine for MINTR0
                000024D0 MIN1     EQU     *               Service routine for MINTR1
                000024D0 MIN2     EQU     *               Service routine for MINTR2
                000024D0 MIN3     EQU     *               Service routine for MINTR3
0024D0 4E75                       RTS



0024D2 6000FFE2   =24B6 INTTBL  BRA.L   ISBXINT
0024D6 6000FFF6   =24CE         BRA.L   INT7
0024DA 6000FFF2   =24CE         BRA.L   INT6
0024DE 6000FFEE   =24CE         BRA.L   INT5
0024E2 6000FFEA   =24CE         BRA.L   INT4
0024E6 6000FFE6   =24CE         BRA.L   INT3
0024EA 6000FFE2   =24CE         BRA.L   INT2
0024EE 6000FFDE   =24CE         BRA.L   INT1
0024F2 6000FFDA   =24CE         BRA.L   INT0
```

```
                        *       Each of the 256 possible combinations of Multibus
                        *       interrupts is represented by a value that determines the
                        *       interrupt to process.  The value is an index into INTTBL.
                        *       In this example, INT0* has the highest priority and the iSBX
                        *       interrupts have the lowest priority.

0024F6 000408080C0C0C0C INTPRI  DC.B    0*4,1*4,2*4,2*4,3*4,3*4,3*4,3*4 ISBX, INT7*, INT6*,
                                                                         INT5*
0024FE 1010101010101010         DC.B    4*4,4*4,4*4,4*4,4*4,4*4,4*4,4*4 INT4*
002506 1414141414141414         DC.B    5*4,5*4,5*4,5*4,5*4,5*4,5*4,5*4 INT3*
00250E 1414141414141414         DC.B    5*4,5*4,5*4,5*4,5*4,5*4,5*4,5*4
002516 1818181818181818         DC.B    6*4,6*4,6*4,6*4,6*4,6*4,6*4,6*4 INT2*
00251E 1818181818181818         DC.B    6*4,6*4,6*4,6*4,6*4,6*4,6*4,6*4
002526 1818181818181818         DC.B    6*4,6*4,6*4,6*4,6*4,6*4,6*4,6*4
00252E 1818181818181818         DC.B    6*4,6*4,6*4,6*4,6*4,6*4,6*4,6*4
002536 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4 INT1*
00253E 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
002546 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
00254E 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
002556 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
00255E 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
002566 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
00256E 1C1C1C1C1C1C1C1C         DC.B    7*4,7*4,7*4,7*4,7*4,7*4,7*4,7*4
002576 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4 INT0*
00257E 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
002586 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
00258E 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
002596 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
00259E 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025A6 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025AE 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025B6 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025BE 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025C6 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025CE 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025D6 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025DE 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025E6 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4
0025EE 2020202020202020         DC.B    8*4,8*4,8*4,8*4,8*4,8*4,8*4,8*4


0025F6 4E71             SBXTBL  NOP                     No iSBX interrupt
0025F8 4E75                     RTS
0025FA 6000FED4   =24D0         BRA.L   MIN0
0025FE 6000FED0   =24D0         BRA.L   MIN1
002602 6000FECC   =24D0         BRA.L   MIN2
002606 6000FEC8   =24D0         BRA.L   MIN3


                        *       This table serves the same purpose as INTPRI but
                        *       applies to iSBX interrupts.

00260A 000408080C0C0C0C SBXPRI  DC.B    0*4,1*4,2*4,2*4,3*4,3*4,3*4,3*4
002612 1010101010101010         DC.B    4*4,4*4,4*4,4*4,4*4,4*4,4*4,4*4
```

```
                    **      MULTI - GENERATING MULTIBUS INTERRUPTS
                    *
                    *       J9, J31 and the GENIRQ bit in the control/status
                    *       register can be used to generate interrupts on
                    *       the Multibus.
                    *
                    *       A Multibus interrupt is generated when SYSIRQ goes
                    *       from 0 to 1.  The interrupt is cleared when the
                    *       interrupted board writes a $0001 to the MPU-28's
                    *       mailbox address.
                    *
                    *       The MPU-28 board is assumed to be configured
                    *       as follows:
                    *
                    *       JUMPER J9
                    *         2 to 3    Interrupt cleared via the mailbox
                    *
                    *       JUMPER J31
                    *         F1 to F2  Interrupt INT5* on the Multibus
                    *

00261A 0039000203F80901 MULTI   OR.B    #SYSIRQ,SCR1     Generate interrupt
002622 023900FD03F80901         AND.B   #$FF-SYSIRQ,SCR1 Remove the interrupt
00262A 4E75                     RTS
```

```
                        **      MULTIA - GENERATING MULTIBUS INTERRUPTS
                        *
                        *       This routine performs the same function as
                        *       MULTI except that the board receiving the
                        *       interrupt clears ACKFLAG in shared memory to
                        *       indicate that the interrupt has been received.
                        *
                        *       The MPU-28 board is assumed to be configured
                        *       as follows:
                        *
                        *       JUMPER J9
                        *         1 to 2    Interrupt cleared via GENIRQ
                        *
                        *       JUMPER J31
                        *         F1 to F2  Interrupt *INT5 on the Multibus
                        *

00262C 31FCFFFF100C     MULTIA  MOVE.W  #$FFFF,ACKFLAG   Set interrupt acknowledge flag
002632 0039000203F80901         OR.B    #SYSIRQ,SCR1     Generate interrupt

00263A 4A78100C         MULTI2  TST.W   ACKFLAG
00263E 66FA       =263A         BNE     MULTI2           Wait for interrupt acknowledge

002640 023900FD03F80901         AND.B   #$FF-SYSIRQ,SCR1 Remove interrupt from Multibus
002648 4E75                     RTS
```

```
                    **      TRANSMAP - SET UP TRANSPARENT TRANSLATION MAP
                    *
                    *       The specified map is configured with a one-to-one
                    *       translation map.  In other words, the 64K blocks of
                    *       dual-ported memory appear on the Multibus in the
                    *       same order they appear in memory.
                    *
                    *       ENTRY (D0.B) = MAP NUMBER (0-3)
                    *             (D2.W) = NUMBER OF 64K BLOCKS OF DUAL PORTED MEMORY
                    *                      FOR EXAMPLE D2=16 FOR 1MB OF MEMORY
                    *

            0000264A TRANSMAP EQU    *
00264A 48E7E080                  MOVEM.L D0-D2/A0,-(A7)
00264E 41F903F80001              LEA     MAP,A0
002654 E148                      LSL.W   #8,D0
002656 E348                      LSL.W   #1,D0
002658 D0C0                      ADD.W   D0,A0            Get address of selected map
00265A 5342                      SUB.W   #1,D2            Create bit mask

00265C 323C0200                  MOVE.W  #MAPLEN/4,D1
002660 4280                      CLR.L   D0
002662 1080              STRA2   MOVE.B  D0,(A0)          Set up translation map
002664 5448                      ADD.W   #2,A0
002666 5240                      ADD.W   #1,D0
002668 C042                      AND.W   D2,D0
00266A 5541                      SUB.W   #2,D1
00266C 66F4       =2662          BNE     STRA2

00266E 4CDF0107                  MOVEM.L (A7)+,D0-D2/A0
002672 4E75                      RTS
```

```
                        **      SETMAP - SETUP TRANSLATION MAP ENTRY
                        *
                        *       Given a Multibus address and an on-board address, this
                        *       routine creates an entry in the translation map to
                        *       perform the required translation.  This routine assumes
                        *       that the actual Multibus address of the board is not
                        *       known and therefore maintains multiple copies of the
                        *       translation map.
                        *
                        *       ENTRY (D0.B) = MAP NUMBER (0-3)
                        *             (D2.W) = NUMBER OF 64K BLOCKS OF DUAL PORTED MEMORY
                        *                      FOR EXAMPLE D2=16 FOR 1MB OF MEMORY
                        *             (A0)   = ON-BOARD ADDRESS
                        *             (A1)   = OFF-BOARD ADDRESS
                        *

002674 48E7E0C0         SETMAP  MOVEM.L D0-D2/A0/A1,-(A7)
002678 2209                     MOVE.L  A1,D1           Save off-board address
00267A 43F903F80001             LEA     MAP,A1
002680 E148                     LSL.W   #8,D0
002682 E348                     LSL.W   #1,D0
002684 D2C0                     ADD.W   D0,A1           Get address of selected map

002686 4841                     SWAP    D1              Use high bits of off-board address
002688 5342                     SUB.W   #1,D2           Create bit mask
00268A C242                     AND.W   D2,D1           Find offset within dual-ported
                                                        memory
 0268C E349                     LSL.W   #1,D1           Scale offset because map is in odd
                                                        bytes only
00268E D2C1                     ADD.W   D1,A1           Adjust map address

002690 2008                     MOVE.L  A0,D0
002692 4840                     SWAP    D0              Use high bits of on-board address
002694 5242                     ADD.W   #1,D2
002696 E34A                     LSL.W   #1,D2
002698 4281                     CLR.L   D1
00269A 13801000         SETMAP2 MOVE.B  D0,(A1,D1)      Set on-board address in
                                                        translation map
00269E D242                     ADD.W   D2,D1
0026A0 B27C0200                 CMP.W   #$0200,D1
0026A4 66F4       =269A         BNE     SETMAP2
0026A6 4CDF0307                 MOVEM.L (A7)+,D0-D2/A0/A1
0026AA 4E75                     RTS

                                END

NO ERRORS
```

------ Notes ------

7. MPU-28 SUPPORT INFORMATION

This section contains pin assignments for Multibus P1 and P2 connectors, I/O headers, iSBX connector, the NMI/reset connector and the parts list.

7-1. Multibus P1 and P2 Edge Connector Pin Assignments

Figures 7-1a and 7-1b list the pin assignments of the Multibus P1 and P2 connectors. Only six signals are connected to the P2 connector; they are shown in Figure 7-1a. All other pins of P2 are not connected.

Figure 7-1a: Multibus P2 Pin Assignments

Pin #	Signal
1	PDN
2	PBKUP
55	AD16*
56	AD17*
57	AD14*
58	AD15*

See Section 3-1-2 for more information.

Figure 7-1b: Multibus P1 Pin Assignments

	Component Side Of The Board			Circuit Side Of The Board		
	Mnemonic	Description	Pin	Pin	Mnemonic	Description
	GND	Signal Ground	1	2	GND	Signal Ground
POWER	+5V	+5Vdc	3	4	+5V	+5Vdc
	+5V	+5Vdc	5	6	+5V	+5Vdc
SUPPLIES	+12V	+12Vdc	7	8	+12V	+12Vdc
	-5V	-5Vdc	9	10	-5V	-5Vdc
	GND	Signal Ground	11	12	GND	Signal Ground
	BCLK*	Bus Clock	13	14	INIT*	Initialize
BUS	BPRN*	Bus Priority In	15	16	BPRO*	Bus Priority Out
	BUSY*	Bus Busy	17	18	BREQ*	Bus Request
CONTROLS	MRDC*	Memory Read Cmd	19	20	MWTC*	Memory Write Cmd
	IORC*	I/O Read Cmd	21	22	IOWC*	I/O Write Cmd
	XACK*	Xfer Acknowledge	23	24	INH1	Inhibit RAM
	LOCK*	Lock	25	26	INH2	Inhibit ROM
BUS	BHEN*	Byte High Enable	27	28	AD10*	
CONTROLS	CBRQ*	Common Bus Request	29	30	AD11*	Address
AND	CCLK*	Constant Clock	31	32	AD12*	Bus
ADDRESS	INTA*	Interrupt Ack	33	34	AD13*	
	INT6*	Parallel	35	36	INT7*	Parallel
INTERRUPTS	INT4*	Interrupt	37	38	INT5*	Interrupt
	INT2*	Requests	39	40	INT3*	Requests
	INT0*		41	42	INT1*	
	ADRE*		43	44	ADRF*	
	ADRC*		45	46	ADRD*	
	ADRA*	Address	47	48	ADRB*	Address
ADDRESS	ADR8*	Bus	49	50	ADR9*	Bus
	ADR6*		51	52	ADR7*	
	ADR4*		53	54	ADR5*	
	ADR2*		55	56	ADR3*	
	ADR0*		57	58	ADR1*	
	DATE*		59	60	DATF*	
	DATC*		61	62	DATD*	
	DATA*		63	64	DATB*	
DATA	DAT8*	Data	65	66	DAT9*	Data
	DAT6*	Bus	67	68	DAT7*	Bus
	DAT4*		69	70	DAT5*	
	DAT2*		71	72	DAT3*	
	DAT0*		73	74	DAT1*	
	GND	Signal Ground	75	76	GND	Signal Ground
POWER	--	Reserved	77	78	--	Reserved
	-12V	-12Vdc	79	80	-12V	-12Vdc
SUPPLIES	+5V	+5Vdc	81	82	+5V	+5Vdc
	+5V	+5Vdc	83	84	+5V	+5Vdc
	GND	Signal Ground	85	86	GND	Signal Ground

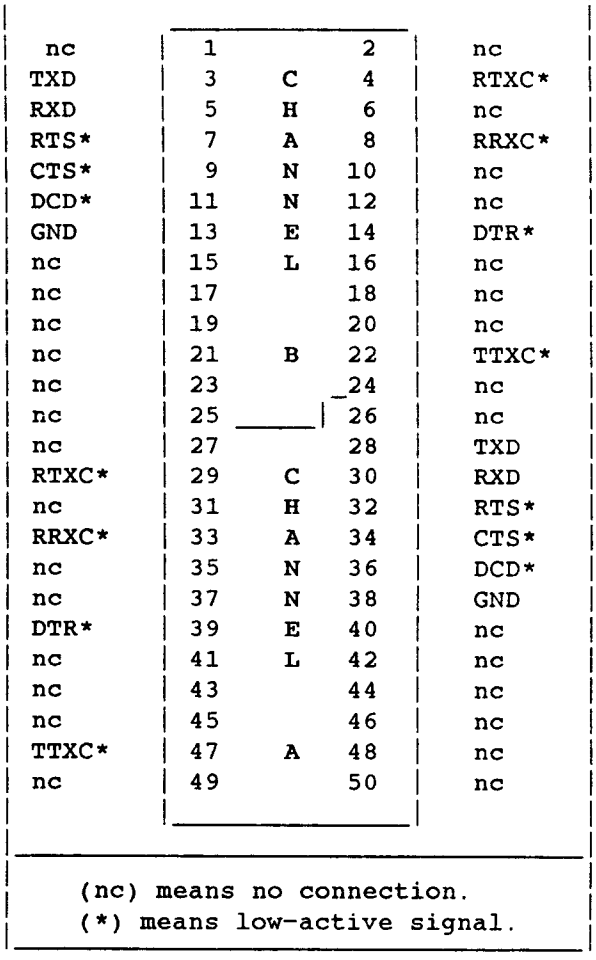
NOTES: Odd-numbered pins are on the component side of the board. Pin 1 is the left-most pin when viewed from the component side of the board with the extractors at the top. All unassigned pins are reserved.

7-2. J6: 50-Pin I/O Header

J6, on the top edge of the MPU-28, provides a connection for two RS-232-C serial devices to the MPU-28.

The MPU-28 is designed to connect easily to a DCE (Data Circuit-Terminating Equipment), or DTE (Data Terminal Equipment) device. The following figures show the layout of signals at the J6 header and their distribution using a 50-conductor ribbon cable and two 25-pin D-type connectors. This can be ordered using Winchester Electronics part number 49-1125P or equivalent; to order the cable from SBE, specify part number CBL-68K10.

Figure 7-2a: MPU-28 As DTE -- J6 Layout & RS-232-C Pin Assignments When Both MPU-28 Ports Are Set Up To Connect To A Modem (Or Other DCE)



nc	1		2	nc
TXD	3	C	4	RTXC*
RXD	5	H	6	nc
RTS*	7	A	8	RRXC*
CTS*	9	N	10	nc
DCD*	11	N	12	nc
GND	13	E	14	DTR*
nc	15	L	16	nc
nc	17		18	nc
nc	19		20	nc
nc	21	B	22	TTXC*
nc	23		24	nc
nc	25		26	nc
nc	27		28	TXD
RTXC*	29	C	30	RXD
nc	31	H	32	RTS*
RRXC*	33	A	34	CTS*
nc	35	N	36	DCD*
nc	37	N	38	GND
DTR*	39	E	40	nc
nc	41	L	42	nc
nc	43		44	nc
nc	45		46	nc
TTXC*	47	A	48	nc
nc	49		50	nc

(nc) means no connection.
(*) means low-active signal.

Figure 7-2b: MPU-28 As DCE -- J6 Layout & RS-232-C Pin Assignments When Both MPU-28 Ports Are Set Up To Connect To A Terminal (Or Other DTE)

nc	1		2	nc
RXD	3	C	4	RTXC*
TXD	5	H	6	nc
CTS*	7	A	8	RRXC*
RTS*	9	N	10	nc
DTR	11	N	12	nc
GND	13	E	14	DCD*
nc	15	L	16	nc
nc	17		18	nc
nc	19		20	nc
nc	21	B	22	TTXC*
nc	23		24	nc
nc	25		26	nc
nc	27		28	RXD
RTXC	29	C	30	TXD
nc	31	H	32	CTS*
RRXC*	33	A	34	RTS*
nc	35	N	36	DTR*
nc	37	N	38	GND
DCD*	39	E	40	nc
nc	41	L	42	nc
nc	43		44	nc
nc	45		46	nc
TTXC*	47	A	48	nc
nc	49		50	nc

(nc) means no connection.
(*) means low-active signal.

Figure 7-2c: J6: RS-232-C Layout When MPU-28 Set Up* For Connection To Modem (Or Other DCE)

D-Type Connector Pin Layout	Signal Direction On 50-Conductor Ribbon Cable (Both Halves)	Ch. B J6 Pin	Ch. A J6 Pin	Schematic Mnemonic For J6 Signal
1		1	26	nc
14		2	27	nc
2	<----------	3	28	TXD
15	---------->	4	29	RTXC*
3	---------->	5	30	RXD
16		6	31	nc
4	<----------	7	32	RTS*
17	---------->	8	33	RRXC*
5	---------->	9	34	CTS*
18		10	35	nc
6	---------->	11	36	DCD*
19		12	37	nc
7	----------	13	38	GND
20	<----------	14	39	DTR*
8		15	40	nc
21		16	41	nc
9		17	42	nc
22		18	43	nc
10		19	44	nc
23		20	45	nc
11		21	46	nc
24	<----------	22	47	TTXC*
12		23	48	nc
25		24	49	nc
13		25	50	nc

J6 is set up for using an unmodified 50-conductor ribbon cable to connect the MPU-28 to two DCE devices such as modems or computers.

* Channel A requires appropriate J14, J15, J16, J17, and J18 setup. Channel B requires appropriate J8, J10, J11, J12, and J13 setup. "nc" indicates no connection.

Figure 7-2d: J6: RS-232-C Layout When MPU-28 Set Up* For Connection To Terminal Or Other DTE

D-Type Connector Pin Layout	Signal Direction On 50-Conductor Ribbon Cable (Both Halves)	Ch. B J6 Pin	Ch. A J6 Pin	Schematic Mnemonic For J6 Signal
1		1	26	nc
14		2	27	nc
2	---------->	3	28	RXD
15	---------->	4	29	RTXC*
3	<----------	5	30	TXD
16		6	31	nc
4	---------->	7	32	CTS*
17	---------->	8	33	RRXC*
5	<----------	9	34	RTS*
18		10	35	nc
6	<----------	11	36	DTR*
19		12	37	nc
7	----------	13	38	GND
20	---------->	14	39	DCD*
8		15	40	nc
21		16	41	nc
9		17	42	nc
22		18	43	nc
10		19	44	nc
23		20	45	nc
11		21	46	nc
24	<----------	22	47	TTXC*
12		23	48	nc
25		24	49	nc
13		25	50	nc

J6 is set up for using an unmodified 50-conductor ribbon cable to connect the MPU-28 to two DTE devices such as terminals.

* Channel A requires appropriate J14, J15, J16, J17, and J18 setup. Channel B requires appropriate J8, J10, J11, J12, and J13 setup. "nc" indicates no connection.

7-3. J7: 34-Pin Parallel I/O Header

The J7 header connector provides connections to the parallel I/O and control lines of the 68230 Parallel Interface/Timer (PI/T). The following figure shows the pin assignments for J7.

Figure 7-3: Pin Assignments Of Connector J7

Pin	Signal	Pin	Signal
1	PA0	2	PA1
3	PA2	4	PA3
5	PA4	6	PA5
7	PA6	8	PA7
9	H1	10	H2
11	GND	12	GND
13	H3	14	H4
15	PB0	16	PB1
17	PB2	18	PB3
19	PB4	20	PB5
21	PB6	22	PB7
23	GND	24	GND
25	PC4	26	PC2
27	PC1	28	PC0
29	GND	30	+5V*
31	GND	32	+5V*
33	GND	34	+5V*

* Fused

NOTE: Fuse F1 is included to protect the MPU-28 in the event that the +5V lines on the parallel port I/O header are shorted to ground. Check this fuse if the corresponding LED (tenth red LED from the green LED) is not lit. See the description of LED programming in Section 3-3.

7-4. J20: 44-Pin iSBX Connector

In the following tables, an "M" indicates Multimodule. For example, "M Data Bit 8" is the Multimodule's Data Bit 8.

Figure 7-4: Pin Assignments For iSBX Connector J20

Pin #	MPU-28 Signal	iSBX Signal/Description		Pin #	MPU-28 Signal	iSBX Signal/Description	
			Left Section				
43	ID8	MD8	M Data Bit 8	44	ID9	MD9	M Data Bit 9
41	ID10	MDA	M Data Bit A	42	ID11	MDB	M Data Bit B
39	ID12	MDC	M Data Bit C	40	ID13	MDD	M Data Bit D
37	ID14	MDE	M Data Bit E	38	ID15	MDF	M Data Bit F
			Right Section				
35	GND	GND	Signal GND	36	+5V	+5V	+5V
33	ID0	MD0	M Data Bit 0	34	nc	MDRQT	M DMA Request
31	ID1	MD1	M Data Bit 1	32	nc	MDACK/M	DMA Acknowledge
29	ID2	MD2	M Data Bit 2	30	nc	OPT0	Option 0
27	ID3	MD3	M Data Bit 3	28	nc	OPT1	Option 1
25	ID4	MD4	M Data Bit 4	26	nc	TDMA	Terminate DMA
23	ID5	MD5	M Data Bit 5	24	nc		Reserved
21	ID6	MD6	M Data Bit 6	22	MCS0*	MCS0*	M Chip Select 0
19	ID7	MD7	M Data Bit 7	20	MCS1*	MCS1*	M Chip Select 1
17	GND	GND	Signal Ground	18	+5V	+5V	+5V
15	IRD*	IORD*	I/O READ Cmnd	16	MWAIT0*	MWAIT*	M Wait
13	IWD*	IOWRT*	I/O WRITE Cmnd	14	MINTR0	MINTR0	M Interrupt 0
11	A1	MA0	M Address 0	12	MINTR1	MINTR1	M Interrupt 1
9	A2	MA1	M Address 1	10	A4	MA4	M Address 4
7	A3	MA2	M Address 2	8	MPST0*	MPST*	M Present
5	RESET	RESET	Reset	6	TCLK	MCLK	M Clock
3	GND	GND	Signal Ground	4	+5V	+5V	+5 Volts
1	+12V	+12V	+12 Volts	2	-12V	-12V	-12 Volts

7-5. J43: SCSI Bus Connector

The SCSI bus interface connector, J43, is a 50-pin male connector (2x25) on .100" centers. The following figure shows the pin assignments for J43.

Figure 7-5: J43: 50-Pin SCSI Bus Connector Pinout

Pin	Signal	Definition	Pin	Signal	Definition
2	DATA0*	Data Bit 0	1	GND	Ground
4	DATA1*	Data Bit 1	3	GND	Ground
6	DATA2*	Data Bit 2	5	GND	Ground
8	DATA3*	Data Bit 3	7	GND	Ground
10	DATA4*	Data Bit 4	9	GND	Ground
12	DATA5*	Data Bit 5	11	GND	Ground
14	DATA6*	Data Bit 6	13	GND	Ground
16	DATA7*	Data Bit 7	15	GND	Ground
18	DATAP*	Data Parity	17	GND	Ground
20	Unused		19	GND	Ground
22	Unused		21	GND	Ground
24	Unused		23	GND	Ground
26	Unused		25	GND	Ground
28	Unused		27	GND	Ground
30	Unused		29	GND	Ground
32	ATN*	Output from the SCSI initiator indicating a message transfer to the target	31	GND	Ground
34	Unused		33	GND	Ground
36	BSY*	Output from the SCSI target indicating that it has been selected	35	GND	Ground
38	ACK*	Output from the SCSI initiator which completes the transfer handshake sequence	37	GND	Ground
40	RST*	Output from a SCSI initiator which resets all devices on the SCSI bus	39	GND	Ground
42	MSG*	Input to the SCSI module which indicates the command is completed	41	GND	Ground
44	SEL*	Output from the SCSI initiator used to select the target	43	GND	Ground
46	CD*	Output from the SCSI target which defines the transfer as a command or data	45	GND	Ground
48	REQ*	Handshake signal from the SCSI target which requests data transfer	47	GND	Ground
50	IO*	Output from the SCSI target that defines transfer direction	49	GND	Ground

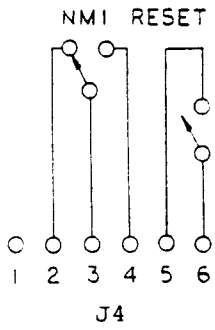
7-6. J4: Reset And NMI Connector

Below are illustrations of the pin assignments of J4 and the wiring of J4 for reset and NMI (Non-Maskable Interrupt).

Figure 7-6a
Pin Assignments Of Connector J4

Pin #	Function
1	+5V for power indicator, etc.
2	NMI switch normally closed
3	Ground for NMI switch
4	NMI switch normally open
5	Reset switch normally open
6	Ground for Reset switch

Figure 7-6b
J4: Wiring Diagram For NMI & Reset Switches



7-7. Power-Up Sequence

At power-up, and whenever a reset is issued to the MPU-28 (whether via a reset button or the INIT line on the Multibus), the 68020 reads the supervisor stack pointer and program counter from addresses 0 and 4 respectively. While these four words are being read, circuitry on the MPU-28 board makes the EPROMs installed in sockets U19 & U49 appear starting at location 0. After the first four memory accesses, the EPROMs return to their normal address, $03F00000.

The PROBUG EPROMs, if installed in sockets U19 & U49, will provide the necessary eight bytes for the 68020 to use at power-up.

7-8. MPU-28 Parts List

The following table lists significant parts for the MPU-28. The SBE part number is given for reference; equivalent parts from other manufacturers may be used except for programmed PALs* and GALs. For the MPU-28 quick reference sheet, jumpering worksheet, standard jumper configuration drawing, board layout, and schematics, refer to the 11" x 17" material in this manual.

On-Board Identification	SBE Mfg. Part No.	Description
U13, U20, U22, U24	11201	1MB DRAM SIP (4MB board)
	or	
U13-U14, U20-U25	11201	1MB DRAM SIP (8MB board)
	or	
U13, U20, U22, U24	11202	256K DRAM SIP (1MB board)
	or	
U13-U14, U20-U25	11202	256K DRAM SIP (2MB board)
J20	21305	Connector, iSBX, 44-Pin, Female
J6, J43	21517	Connector, Header, 50-Pin, Male
--	21901	Jumpers, Insulated, .001"
SW1	30105	8-Position Switch
F1	30305	Fuse, 3 Amp
RN9	41664	Resistor, 10K NET SIP, 8-pin
RN6, RN7, RN8	41665	Resistor, 3.3K NET SIP, 8-pin
RN1, RN11	41668	Resistor, 3.3K NET SIP, 10-pin
RN5, RN10, RN12, RN13	41669	Resistor, 10K NET SIP, 10-pin
RN2, RN3, RN4	41685	Resistor, 220/330 NET SIP, 8-pin
C2	43162	Capacitor, 6.8mfd, 35V, Tantalum
C74	43166	Capacitor, 100mfd, 10V, Tantalum
CR1	44690	Green LED
CR2-CR4	44691	Red LED
U10	48953	Crystal, 7.3728MHz Oscillator
U73	48958	Crystal, 40MHz Oscillator (20MHz board)
	or	
U73	48996	Crystal, 25MHz Oscillator (12.5MHz board)
U74	48992	Crystal, 20MHz Oscillator
U1, U2, U12, U28	54392	RS-232 Driver, 145406P
U58	55403	68020, 32-Bit Microprocessor (20MHz board)
	or	
U58	55458	68020, 32-Bit Microprocessor (12.5MHz board)
U64	55493	68881/68882, FPU (12.5MHz board)
	or	
U64	55496	68881/68882 FPU (20MHz board)
U40	55537	IC, Delay Line, 50ns, triple
U59	55543	IC, Delay Line, 200ns, 5 tap
U38	55544	IC, Delay Line, 90ns
U37	55548	IC, Delay Line, 50ns, 5 tap
U35	55550	IC, Delay Line, 25ns, 5 tap

On-Board Identification	SBE Mfg. Part No.	Description
U82, U95	56131	ICM,CY2148, SRAM 1K X 4
U94, U109-U110, U124	56394	IC, CY7C164, SRAM 16K X 4
U29	66104	IC, PAL16L8
U36	66106	IC, PAL16L8A
U57	66109	IC, PAL20L10A
U50	66110	IC, PAL16R4B
U65	66113	IC, GAL20V8
U78	66114	IC, PAL16L8
U66	66115	IC, PAL16L8
U79	66116	IC, PAL20L8A
U77	66117	IC, PAL20L8A
U56	66118	IC, PAL16L8A
U51	66119	IC, PAL16L8A
U67	66120	IC, PAL20L8A
U11	66246	IC, PAL16R4A
U26	66247	IC, PAL16L8A
U30	66248	IC, PAL16L8
U34	66249	IC, PAL20L8A
U45	66250	IC, PAL20L8B
U52	66251	IC, PAL16L8
U72	66252	IC, PAL16P8A
U93	66267	IC, PAL20L8B
U108	66254	IC, PAL20L8A
U111	66255	IC, PAL16L8A
U16	66263	IC, PAL16L8B
U55	66264	IC, PAL22V10A
U39	66293	IC, PAL16L8 (for 4- and 8MB boards)
	or	
U39	66294	IC, PAL16L8 (for 1- and 2MB boards)
R40-R43	41050S	Resistor, 18 Ohm, 1/8W, 5%
R11-R12, R22, R23, R26-R39	41051S	Resistor, 10 Ohm, 1/8W, 5%
R7, R9, R14, R17, R24-R25, R45-R49	41055S	Resistor, 33 Ohm, 1/8W, 5%
R1, R3-R6, R8, R13, R15-R16, R19-R21, R44	41091S	Resistor, 3.3K, 1/8W, 5%
R18	41103S	Resistor, 10K, 1/8W, 5%
R2	41122S	Resistor, 68K, 1/8W, 5%
Bypass Caps	43151S	Capacitor, .1mfd, 50V, Ceramic
U125, U127, U128, U131	51080S	IC, 74LS74
U33	51096S	IC, 74LS138
U3, U46, U126, U135, U136, U154, U159	51109S	IC, 74LS164
U84, U89, U113	51137S	IC, 74LS240

On-Board Identification	SBE Mfg. Part No.	Description
U96, U153, U158	51139S	IC, 74LS258
U6, U7, U8, U53, U83, U85, U91, U120, U121	51141S	IC, 74LS245
U4, U42, U45, U48, U54, U83, U103, U123	51148S	IC, 74LS273
U98	51178S	IC, 74LS374
U70, U129, U142, U148	51185S	IC, 74LS393
U47	51189S	IC, 74LS640
U100, U101, U115, U116	51191S	IC, 74LS645
U76	51192S	IC, 74LS645-1
U5, U99, U112, U114	51194S	IC, 74LS688
U71	52065S	IC, 74F27
U161	52302S	IC, 74F51
U150, U156	52306S	IC, 74F10
U92	52312S	IC, 74F08
U160	52317S	IC, 74S38
U152	52322S	IC, 74S37
U140	52328S	IC, 74S260
U60, U106, U107	52330S	IC, 74S240
U104, U130, U133, U137, U147, U155, U167	52337S	IC, 74F74
U68, U105, U132, U138, U145, U151, U162	52339S	IC, 74F00
U143	52340S	IC, 74F174
U49	52344S	IC, 74F273
U41, U134	52345S	IC, 74F02
U69, U118, U146, U149	52348S	IC, 74F280
U43, U44	52349S	IC, 74F240
U61, U62, U119, U139, U141, U157, U163, U164, U165, U166	52351S	IC, 74F258
U80	52353S	IC, 74F245
U144	52354S	IC, 74F04
U63	52359S	IC, 74F521
U86-U88, U90, U102, U117	53418S	IC, 74ALS648
U15	55414S	IC, 53C90
U27	55402S	IC, 8530, Serial Communications Controller
U17	55478S	IC, 68230, Parallel Interface/Timer
U9	55500S	IC, 555
U75, U97	59189S	IC, 74HCT640
U81, U122	523605S	IC, 74F640

APPENDIX A: TECHNICAL BULLETINS

This section is reserved for Technical Bulletins. See the following pages for the latest information regarding the MPU-28 board.

If you receive a Technical Bulletin in the mail concerning this product, please update the appropriate pages of this manual and/or store the document in this appendix.

------ NOTES ------

APPENDIX B: SCHEMATICS AND DRAWINGS

Included in this manual are 11 x 17 drawings to aid you in configuring your board. The quick reference sheet briefly describes the general function and standard jumpering of all the jumper areas and connectors on the MPU-28. The jumpering worksheet follows which you can use with the standard jumper configuration drawing to record your board's configuration. Also included is the board layout and schematics for the MPU-28.

B-1. Title Of Schematic Pages

------ NOTES ------

APPENDIX C: SUPPLEMENTARY MANUALS

These manuals and data sheets appear after the 11 x 17 sheets in this manual. If you want to photocopy any of this material, you must obtain written permission from the company it came from.

8530 Serial Communications Controller (SCC)
* Contact: Advanced Micro Devices, Inc.
  901 Thompson Place
  P.O. Box 3453
  Sunnyvale, CA 94088-3000

68230 Parallel Interface/Timer (PI/T)
* Contact: Motorola, Inc.
  3501 Ed Blustein Blvd.
  P.O. Box 6000
  Austin, TX 78762

53C90 SCSI Enhanced Processor (SCSI)
* Contact: NCR Corporation
  1635 Aeroplaza Drive
  Colorado Springs, Co 80916

68881 Floating-Point Coprocessor
* See information for the Motorola chip 68230

An excellent source of information on the optional 68881/68882 floating-point coprocessor is available from Motorola:

MC68881/MC68882 Floating-Point Coprocessor User's Manual, 1987 (First Edition)